MAIKEL JEAN KLEIN CHRIST

RUDINEI LUIS PETTER

AVALIAÇÃO DO DESEMPENHO DOS PROTOCOLOS BONDING E MPTCP EM INSTÂNCIAS LXC E KVM COM NUVEM OPENNEBULA

Três de Maio

2018

MAIKEL JEAN KLEIN CHRIST
RUDINEI LUIS PETTER

AVALIAÇÃO DO DESEMPENHO DOS PROTOCOLOS BONDING E MPTCP EM INSTÂNCIAS LXC E KVM COM NUVEM OPENNEBULA

Trabalho de Conclusão de Curso
Sociedade Educacional Três de Maio
Faculdade Três de Maio
Tecnologia de Redes de Computadores

Orientador:
Prof. Dr. Dalvan Jair Griebler

Três de Maio
2018

RESUMO

O estudo dos protocolos *Bonding* e MPTCP trazem como benefício o aproveitamento da infraestrutura já existente, utilizando-se da total capacidade que o *hardware* oferece para obter maior desempenho e redundância a falhas. O principal objetivo deste trabalho é estudar, compreender, implantar os protocolos *Bonding* e MPTCP em nuvem privada com instancias LXC e KVM para identificar quais dos protocolo consegue atingir o melhor desempenho em um ambiente de nuvem OpenNebula. Os resultados alcançados demonstram que a aplicação do protocolo *bonding* tem o mesmo comportamento em ambientes de nuvem com instâncias LXC e KVM. O protocolo MPTCP apresentou resultados próximos ao ambiente nativo somente em instâncias KVM, havendo degradação de desempenho na rede quando aplicado em ambiente de nuvem com instâncias LXC.

Palavras-chave: Redes de Computadores, Computação em Nuvem, MPTCP, *Bonding*, LXC, KVM.

ABSTRACT

The study of the Bonding and MPTCP protocols have as a benefit the use of existing infrastructure, using the total capacity that the hardware offers for increasing performance and redundancy to failures. The main objective of this work is to study, understand, implement the private cloud with the Bonding and MPTCP protocols over LXC and KVM instances to identify which protocols can achieve the best performance in an OpenNebula cloud environment. The results demonstrated that the application of the bonding protocol has the same behavior in cloud environments with LXC and KVM instances. The MPTCP protocol presented results close to the native environment only in KVM instances, with degradation of performance in the network when applied in a cloud environment with LXC instances.

Keywords: Computer Networks, Cloud Computing, MPTCP, Bonding, LXC, KVM.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Programming Interface*
ARP	*Address Resolution Protocol*
CPU	*Central Processing Unit*
CSC	Cliente do Serviço em Nuvem
CSMA	*Carrier Sense Multiple Access*
DFS	*Distributed File System*
ECMP	*Equal Cost Multipath*
EMPTCP	*Efficient Multipath Transmission Control Protocol*
FDMA	*Frequency Division Multiple Access*
GPU	*Graphics Processing Unit*
IaaS	*Infrastructure as a Service*
IBM	*International Business Machines*
ICMP	*Internet Control Message Protocol*
IEEE	*Institute of Electrical and Electronic Engineers*
IEC	*International Electrotechnical Commission*
IETF	*Internet Engineering Task Force*
IP	*Internet Protocol*
IPV4	*Internet Protocol version 4*
ISO	*International Organization for Standardization*
KVM	*Kernel Virtual Machine*
LACP	*Link Aggregation Control Protocol*
LACPDU	*Link Aggrega-tion Control Protocol Data Unit*
LARCC	*Laboratory of Advanced Researches on Cloud Computing*

LXC	*Linux Containers*
MDLA	*Morphological Dynamic Link Architecture*
MLT	*Multi-Link Trunking*
MPTCP	*MultiPath Transmission Control Protocol*
PaaS	*Platform as a Service*
QoS	*Quality of Service*
RAM	*Random Access Memory*
SDN	*Softwaere Defined Network*
SETREM	*Sociedade Educacional Três de Maio*
SLA	*Service Level Agreement*
SLOs	*Service Level Objectives*
SO	*Sistema Operacional*
SQL	*Structured Query Language*
TCP	*Transmission Control Protocol*
TI	Tecnologia da Informação
VCAT	*Virtual Concatenation*
VM	*Virtual Machine*
VMware	*Virtual Machine Ware*
VPC	*Virtual Private Cloud*
XEN	*Xenoserver*

SUMÁRIO

INTRODUÇÃO

Nos paradigmas atuais, novas tendências tecnológicas surgem com o objetivo de prover serviços sob demanda no ramo da tecnologia da informação, pagando somente pelo uso. Uma ferramenta que oferece o pagamento sob-demanda é a computação em nuvem, a qual fornece a quaisquer usuários hospedagem e terceirização da demanda de TI necessária. A demanda oferecida trata-se de provisionamento de *hardware*, desenvolvimento de aplicações e hospedagem de aplicações.

Para suprir esta demanda é exigido cada vez mais agilidade, eficiência e flexibilidade. Os estudos realizados pela tendencia mundial de *Green IT* ou TI Verde vêm trazendo otimização dos recursos já oferecidos pelos provedores de serviços em nuvem. Existem várias técnicas voltadas para explorar todo o potencial da rede.

Conforme Aust et al. (2006) a agregação de *link*, consiste em uma solução que combina vários métodos de múltiplas conexões de redes paralelas em um único *link* lógico melhorando a capacidade da rede, aumentando a largura de banda e oferecendo redundância. Isso acontece devido a agregação das interfaces de rede em um único caminho, somando a largura de banda e proporcionando tolerância a falhas pelo fato de interrupção do serviço de uma interface não comprometer a disponibilidade do *link*.

Visando agregação de link, um dos protocolos de agregação de *link* é o *bonding*, o qual elimina a ociosidade do *hardware* de rede combinando várias interfaces de rede em uma única interface lógica, oferecendo redundância e otimizando custos. Já a utilização do protocolo (*Multipath TCP*), reduz o tempo de transmissão dos pacotes, distribuindo-os em vários sub-fluxos, e reduzindo a perda dos pacotes.

Para avaliar os protocolos mencionados acima, a utilização do modelo em ambiente nativo se faz necessário para parametrizar os resultados, pelo fato desta aplicação ser executada diretamente sobre o *hardware*, oferecendo maior agilidade na execução do sistema, não havendo a abstração oferecida pela virtualização.

Com a finalidade de implantar os testes em um modelo de implementação em nuvem, a ferramenta OpenNebula é um alternativa de código aberto para nuvens IaaS, que pode utilizar-se de várias máquinas virtuais junto á *data centers*.

Na nuvem computacional OpenNebula, a utilização do LXC (*Linux Container*) oferece característica de uma máquina virtual, são executadas de um *kernel* e tem simulação de *hardware*.

A utilização de instâncias KVM (*Kernel Virtual Machine*) tem como principal característica atuar como um *kernel* a parte dentro do sistema operacional nativo, oferecendo toda a infraestrutura para execução.

Os protocolos *Bonding* e MPTCP, que serão implantados em servidores gerenciados pela ferramenta OpenNebula, com instâncias LXC e KVM, tem o principal objetivo de avaliar quais dos protocolos oferece melhor desempenho na agregação de *links*.

O trabalho tem como objetivo dar continuidade aos estudos do Artigo Rista et al. (2017), o qual aborda o alto desempenho do ambiente em nuvem, implantando instâncias baseadas em contêiner, nas quais os aplicativos do Hadoop podem ser executados.

As contribuições do artigo Artigo Rista et al. (2017), estão descritas a seguir: Analisar e coletar os resultados adquiridos pela aplicação do protocolo MPTCP em ambiente nativo e em instâncias KVM e LXC em nuvem computacional OpenNebula. Analisar e coletar os resultados adquiridos pela aplicação do protocolo *bonding* com instâncias KVM.

A estrutura deste trabalho consiste em 3 Capítulos. O Capítulo 1 descreve aspectos metodológicos, delimitação do tema, objetivos, hipóteses, e períodos sobre a realização dos estudo. No Capítulo 2 estão apresentados as revisões literárias, abordando assuntos como Computação em Nuvem, Agregação de Links, MPTCP e Bonding, sendo definições importantes para a realização deste trabalho. Por fim, o Capítulo 3 que trás os resultados adquiridos com os estudos.

CAPÍTULO 1: PROJETO DE PESQUISA

1.1 TEMA

Avaliação de desempenho dos protocolos *Bonding* e MPTCP (*Multi-Path TCP*) em ambiente de computação em nuvens *OpenNebula* com instâncias LXC (*Linux Containers*), KVM (*Kernel-based Vitual Machine*) e ambiente nativo.

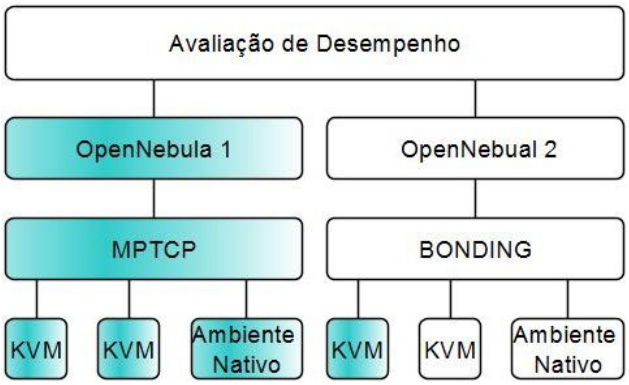
1.1.1 Delimitação do Tema

Esta pesquisa busca avaliar diferentes tecnologias de agregação de *link*, *Bonding* e MPTCP, em ambientes de nuvem privada, diferenciando-os no quesito de desempenho. Os testes serão realizados em uma nuvem computacional *OpenNebula* com instâncias LXC, KVM e ambiente nativo. O estudo será realizado no Laboratório de Pesquisas Avançadas para Computação em Nuvem *(LARCC)* na SETREM.

Conforme demonstrado na Figura 1, os retângulos em branco tratam de assuntos citados no trabalho Rista et al. (2017). Já em cinza, está disposto a relação de novos testes a serem realizados como extensão da pesquisa citada acima. O projeto será realizado pelos acadêmicos Maikel Jean Klein Christ e Rudinei Luis Petter do curso de Tecnologia em Redes de Computadores da SETREM, na área específica de Computação em Nuvem no período de Agosto de 2017 e Junho de 2018.

É importante destacar que este trabalho irá contribuir com um novo con-

Figura 1.1: Descrição do Escopo do Trabalho.



junto de experimentos, possibilitando um investigação do desempenho das soluções de agregação de *link* para ambientes de computação em nuvem privada. Além de reproduzir experimentos anteriores do trabalho Rista et al. (2017), novos experimentos e uma análise comparativa será realizada, conforme destacado no Figura 1.1.

1.2 OBJETIVOS

1.2.1 Objetivo Geral

O objetivo geral é avaliar e comparar o desempenho dos protocolos *Bonding* e *MPTCP* em ambientes de computação em nuvem.

1.2.2 Objetivos Específicos

Os objetivos específicos deste trabalho são os seguintes:

- Aprender os protocolos *Bonding* e *MPTCP*, bem como computação em nuvem com OpenNebula.
- Discutir sobre os trabalhos anteriores e relacionados a este tema.
- Avaliar e comparar o desempenho dos protocolos *Bonding* e MPTCP em ambientes de nuvem privada (LXC e KVM).

- Publicar artigos sobre o estudo realizado.

1.3 JUSTIFICATIVA

O processamento de grandes volumes de dados (*Big Data*) e o armazenamento de dados distribuídos vem gerando um uso intensivo da rede. Com isso, é necessário o uso alternativo de tecnologias de rede para o aumento da largura de banda. Isso pode ser feito de diferentes formas em nível de software e hardware.

Esta solução é a agregação de *link*, que implementa vários métodos para combinar múltiplas conexões de rede. Isso pode ser usado com a finalidade de aumentar o rendimento ou fornecer redundância em caso de falha em algum *link*. Um exemplo é o método aplicado pelo protocolo *Bonding*, que busca a combinação de uma série de portas físicas para criar um único *link* lógico de dados. Dependendo da quantidade e capacidade dos *links* físicos, este protocolo consegue oferecer um *link* de alta largura de banda, compartilhando a carga de tráfego entre as portas delegadas.

Outro método para aumentar a largura de banda da rede é o *Multipath* TCP ou MPTCP. Ele utiliza a agregação de duas ou mais rotas, formando somente um *link* para melhorar o desempenho das comunicações nos canais de destino evitando conflitos entre conexões, facilitando a comunicação entre canais e aumentando a velocidade de troca de informações. MPTCP é nada mais que a eficiência da rede, melhorando a comunicação entre a troca de dados entre um cliente e um servidor.

Esta pesquisa buscou trazer para dentro do ambiente de computação em nuvem os benefícios oferecidos pelos protocolos *Bonding* e MPTCP. Dentro destes benefícios se encontra a alta disponibilidade da rede através de redundância a falhas e maior largura de banda, sem que haja a necessidade da aquisição de novos ativos de rede.

O protocolo *Bonding* alcança a alta largura de banda através da agregação das interfaces de rede disponíveis, já o protocolo MPTCP aumenta a largura de banda. Como contribuição científica o nosso trabalho trás os resultados alcançados através da avaliação de desempenho comparando o ambiente de nuvem IaaS OpenNebula com instâncias LXC e KVM ao ambiente nativo.

Neste trabalho, o foco é reaproveitar as interfaces de rede existentes e aumentar a largura de banda com o auxílio de um software que gerencie múltiplos *links*. O uso destas duas tecnologia de agregação, beneficia na avaliação e no desempenho da rede, melhorando a comunicação entre cliente e servidor.

1.4 PROBLEMA

Considerando a delimitação do tema, o problema em questão é responder quais dos protocolos de agregação de *link* oferece o melhor desempenho em um ambiente de nuvem privada com instâncias LXC e KVM?

1.5 HIPÓTESES

- A vazão de rede usando MPTCP e *Bonding* diverge significativamente em instâncias de ambiente de nuvem baseado em LXC.
- A vazão de rede usando MPTCP e *Bonding* diverge significativamente em instâncias de ambiente de nuvem baseado em KVM.
- A latência da rede dos protocolos MPTCP e *Bonding* aumenta significativamente ao implantá-los em ambientes de nuvem com instancias LXC e KVM.

1.6 METODOLOGIA

Esta seção descreve quais métodos serão explorados pelos pesquisadores, a fim de demonstrar quais termos filosóficos e científicos são utilizados para embasar os conceitos deste trabalho.

1.6.1 Método de Abordagem

Segundo Lovato (2013), a abordagem qualitativa não se utiliza de dados numéricos ou dados estatísticos, mas sim de conclusões descritivas, no qual busca-se entender os fenômenos estudados. Para esta pesquisa, foi utilizado o método de abordagem qualitativo, adquirindo conhecimento e buscam entender determinados termos.

Já a abordagem quantitativa busca suas conclusões em dados estatísticos e bases numéricas, na qual os resultados afetam diretamente na decisão das conclusões. Compreendendo o assunto proposto para a pesquisa, afim de buscar maior embasamento para obter um resultado final. Com os resultados adquiridos pela pesquisa, a abordagem quantitativa demonstra os números adquiridos pelos *benchmarks* aplicados estatisticamente, destacando então o desempenho em aplicação de nuvem das duas aplicações.

1.6.2 Métodos de Procedimento

Segundo Lovato (2013), os procedimentos são ordenações dos métodos em conjuntos de atividades, para desenvolver um plano de estudo com maior segurança, identificar erros antecipados, auxiliando com as decisões dos pesquisadores.

Para esta pesquisa serão utilizados dois procedimentos: pesquisa bibliográfica e experimento. Com a finalidade de entender os assuntos, a pesquisa bibliográfica servirá de embasamento para o entendimento do funcionamento de todas as ferramentas utilizadas nesta pesquisa. O estudo de caso foi feito através da implantação dos protocolos *IEEE 802.3ad* e MPTCP, posteriormente realizando testes e verificando qual o protocolo que trará mais benefícios.

1.6.2.1 Pesquisa bibliográfica

O procedimento deste trabalho será embasado em pesquisas bibliográficas, estudando obras escritas por autores renomados, reunindo material obtido através de livros, revistas e publicações.

1.6.2.2 Estudo de Caso

Segundo Lovato (2013), o estudo de caso visa através de uma realidade delimitada obter um conhecimento profundo da pesquisa. No presente trabalho os artigos estudados servirão de embasamento para a aplicação prática, as quais resultarão em métricas de desempenho que serão documentadas provando a veracidade das hipóteses.

1.6.3 Técnica

A técnica levantada e utilizada será através da testagem, com a finalidade de obter dados, verificando e comparando as duas tecnologias com frequência do mesmo padrão, analisando os rendimentos individuais para originar resultados e realizar comparações.

1.6.4 Validação das Hipótese

As hipóteses serão validadas a partir de testes do *Multipath* TCP e *Bonding* em cima do uso de *benchmarks*. Para validar a primeira hipótese que busca provar o desempenho do protocolo MPTCP com instâncias LXC em ambiente de nuvem privada provê menos de 1% no ganho de desempenho no ambiente nativo, e para validar a segunda hipótese que busca provar se o desempenho dos protocolos *Bonding* e MPTCP com instâncias KVM em nuvem privada IaaS acarreta em ganho de 1% em relação ao ambiente nativo.

Serão executados *benchmarks* de avaliação de desempenho de rede, com

foco em *throughput* e latência, sendo realizados nos servidores, no ambiente nativo e em nuvem computacional OpenNebula com instâncias LXC e KVM no LARCC, provando se realmente há melhoras na transmissão de dados utilizando o protocolo MPTCP e *Bonding*.

Já para validar a terceira hipótese, onde em ambientes de nuvem, o protocolo *Bonding* oferece melhores resultados ao ser analisado por métricas de desempenho de rede em relação ao MPTCP, serão executados *benchmarks* nos servidores do LARCC, para posteriormente analisar em qual das duas situações o protocolo melhora os resultados na transmissão de dados.

1.7 ORÇAMENTO

Para a realização do projeto, foi realizado uma tabela identificando os gastos, abaixo o orçamento das despesas com a elaboração do projeto.

Quadro 1.1: Orçamento

Descrição	**Valor Unitário**	**Quantidade**	**Total**
Deslocamento	R$ 15,00	124	R$139,00
Impressão	R$ 0,15	800	R$ 120,00
Encadernação	R$ 2,55	6	R$ 8,55
Preço por Hora Trabalhada	R$ 40,00	30 Semanas (32 horas/semana)= 1280 Horas	R$ 38.300,00
Total			R$ 38.667,55

1.8 CRONOGRAMA

O cronograma é uma etapa importante do projeto de pesquisa, apresentam de forma detalhada todas as etapas de trabalho bem como o período de realização de cada etapa.

- A1: Escolha do Tema.
- A2: Elaboração do Capítulo 1.

- A3: Aprender o conteúdos dos protocolos *Bonding* e MPTCP e documentar no Capítulo 2.
- A4: Aprender sobre a Nuvem OpenNebula e documentar no Capítulo 2.
- A5: Documentação Capitulo 2.
- A6: Estudo sobre *Benchmarks* e documentação no Capítulo 2.
- A7: Discutir os trabalhos anteriores e relacionados ao Tema.
- A8: Criar tabela sobre artigos relacionas e documentar discussão no Capítulo 2.
- A9: Implantar as ferramentas nos ambientes sugeridos.
- A10: Avaliar e comparar o desempenho dos protocolos *Bonding* e MPTPC em ambientes de nuvem privada (LXC e KVM).
- A11: Correções de possíveis problemas.
- A12: Documentação Capítulo 3.
- A13: Conclusão / Considerações Finais.
- A14: Publicar artigo sobre o estudo realizado.

Quadro 1.2: Cronograma

Atividades	2017					2018					
	Ago	Set	Out	Nov	Dez	Jan	Fev	Mar	Abr	Mai	Jun
A1	X	X									
A2	X	X	X	X	X						
A3	X	X	X	X	X	X	X	X			
A4			X	X	X	X	X	X			
A5	X	X	X	X	X	X	X	X			
A6						X	X	X	X	X	X
A7						X	X	X	X	X	X
A8						X	X	X	X	X	X
A9								X	X	X	X
A10								X	X	X	X
A11								X	X	X	X
A12								X	X	X	X
A13										X	X
A14											

CAPÍTULO 2: BASE TEÓRICA

No cenário atual de computação em nuvem, a virtualização dos ambientes computacionais está sendo adotada por cada vez mais instituições ou empresas. O estudo e compreensão destes ambientes colabora diretamente para o conhecimento identificando qual modelo adequa-se a realidade do cliente.

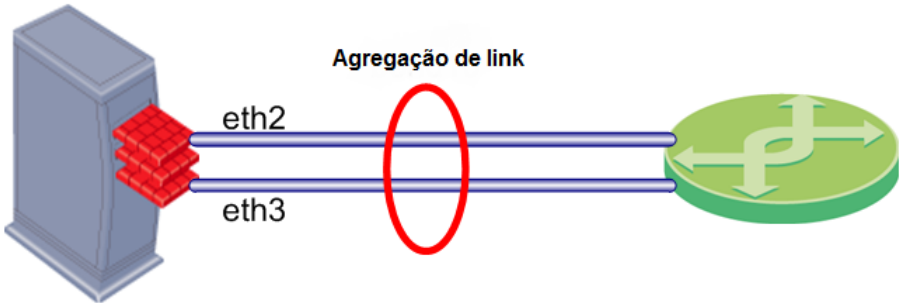
2.1 AGREGAÇÃO DE *LINKS*

Todas as definições que cercam a Agregação de *link* têm por principal característica a combinação de várias interfaces físicas em uma interface lógica. Essa técnica tem por objetivo melhorar a capacidade da rede, aumentando a velocidade de transmissão sem a necessidade de investimentos em mais interfaces, assim, reduzindo custos. A velocidade do *link* lógico consiste na soma das velocidades de todos os *links* físicos utilizados. Por exemplo, quando criado a agregação de quatro *links* de 100 Mbps, a velocidade teórica total seria de 400 Mbps.

A Agregação de *Link* (LAG) é usada para múltiplos enlaces em um *link* lógico, descrevendo vários métodos para o uso de múltiplas conexões de redes, usando laços de repetições para aumentar a taxa de transferência além do limite que um único *link* pode alcançar.

Uma característica importante de agregação de *link* é que quando uma interface de rede falhar, o tráfego desta interface será distribuída entre as outras que fazem parte da agregação, e quando a taxa de transmissão for lenta, pode-se então, a partir de uma configuração específica, priorizar o envio de determinados

Figura 2.1: Agregacao de link



Fonte: Extraído de Hintersteiner (2016)

dados. Usar a agregação de *links* é eficiente para aumentar a largura de banda quando necessário e muito eficiente para economia com a utilização de *hardwares*.

2.1.1 Opções de *Driver* de ligação

Conforme Aust et al. (2006), a implementação do protocolo de agregação é facilmente encontrada em distribuições Linux. A interface de ligação Linux leva o nome de *bond0* sendo configurada com o comando *ifconfig*, fixando um endereço IP e endereço MAC como qualquer outra interface Linux.

Normalmente, o endereço MAC da interface de ligação é da primeira interface ligada. Durante a ligação, cada interface ligada muda seu endereço MAC para o MAC da interface de ligação. Assim, evitando conflito ou perda de pacotes com switches e roteadores configurados com endereço IP relacionados a endereços MAC de interfaces que façam parte da ligação *bond0*.

Atualmente, a implementação suporta 7 modos de configuração diferentes. O usuário pode escolher o modo que melhor lhe convém e configurá-lo. A Tabela 2.1 descreve os modos de operação.

Conforme Davis et al. (2011), os modos que especificam as políticas de vinculações do protocolo *bonding* são:

Balance-rr ou 0: *Round-robin policy*, transmite todos os pacotes em sequên-

Quadro 2.1: Modos de Ligação Linux

Algoritmo *Bonding*	Modo	Descrição
Balance-RR	0	Política Round Robin possibilita a transmissão de pacotes em ordem sequencial entre interfaces com fio.
Active-Backup	1	Política de *Backup* Ativo. A segunda interface é definida para tolerância a falhas.
Balance-XOR	2	Fornece balanceamento de carga e tolerância a falhas.
Broadcast	3	Transmissão de datagrama em cada interface.
802.3ad	4	Padrão de agregação de *link* dinâmico IEEE 802.3ad.
Balance-tlb	5	Balanceamento de carga de transmissão adaptativa.
Balance-alb	6	Balanceamento de carga adaptativo, incluindo *Balance-tlb plus*, balanceamento de carga de recebimento.

Fonte: Extraído de Aust et al. (2006)

cia do primeiro nó escravo até o último, fornecendo balanceamento de carga e tolerância a falhas.

Active-backup ou 1: *Active-backup policy*: somente um escravo permanece ativo, a única possibilidade de outro escravo tornar-se ativo é somente em caso de falha. Neste modo, o endereço *MAC* fica visível em apenas uma porta do adaptador de rede, afim de evitar com que a chave venha a confundir-se entre escravos. Esta configuração oferece mais tolerâncias a falhas.

Balance-xor ou 2: *XOR policy*: Efetua transmissões baseadas na política de *hash*. A política padrão é simples, onde o endereço *MAC* de origem é aplicado *XOR* com o endereço *MAC* destino do pacote *XOR*, assim contando os escravos no módulo. Este módulo fornece balanceamento de carga e tolerância a falhas.

Broadcast ou 3: *Broadcast policy*: permite o trafego de dados em todas as interfaces escravas, o qual fornece tolerância a falhas.

802.3ad ou 4: *IEEE 802.3ad Dynamic link aggregation*: cria grupos de agregação com as mesmas configurações de velocidade e *duplex*. De acordo com as especificações *802.3ad*, é definido quais caminhos serão ativos. A definição de quais caminhos serão responsáveis pelo tráfego de saída é feita conforme a política de transmissão em *hash*, podendo ser alterado a partir da política *XOR X* via *xmit hash policy.*

Nem todas as políticas de transmissão são compatíveis com o protocolo *802.3ad*. Os pré-requisitos começam pelo suporte *Ethtool* nos *drivers* para velocidade e *duplex* de cada escravo e um *switch* que suporte a agregação de *link IEEE 802.3ad*.

Balance-tlb ou 5: *Adaptive transmit load balancing*: efetua a ligação entre os canais que não necessitem de nenhum suporte de alguma chave especial. No *tlb dynamic lb* modo 1 o tráfego de saída é organizado de acordo com a carga atual da rede disposta em cada escravo, levando em consideração a velocidade de tráfego.

Balance-alb ou 6 *adaptative load balancing*: aplicado no padrão *IPV4*, este sistema de balanceando de carga otimiza o trafego sem nenhum suporte de chave especial onde o balanceamento de carga de recepção é alcançado pelo *ARP*.

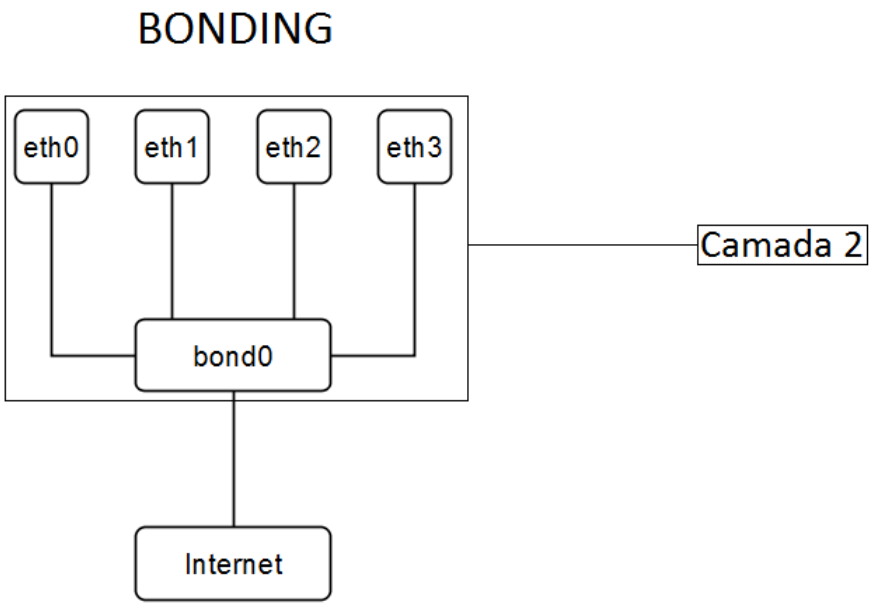
2.1.2 Protocolo *Bonding*

Bonding é um protocolo de agregação de link que tem por principal característica agregar diversas interfaces físicas em uma única interface lógica, aumentando a largura de banda, redundância e disponibilidade do *link*.

Segundo Coutinho et al. (2013), o protocolo *Bonding* atua no controle de camada 2 e pode ser utilizado para detectar automaticamente, configurar, e gerir uma ligação lógica única com várias ligações físicas entre os dois dispositivos adjacentes ativados . Assim, agregação de *link* fornece maior disponibilidade e capacidade, enquanto melhorias de desempenho de rede são obtidos utilizando o

hardware existente. A Figura 2.2 demonstra a arquitetura do protocolo *Bondind*.

Figura 2.2: *Link* Físico e *Link* Lógico.



Fonte: Adaptado de Sarabando (2017a)

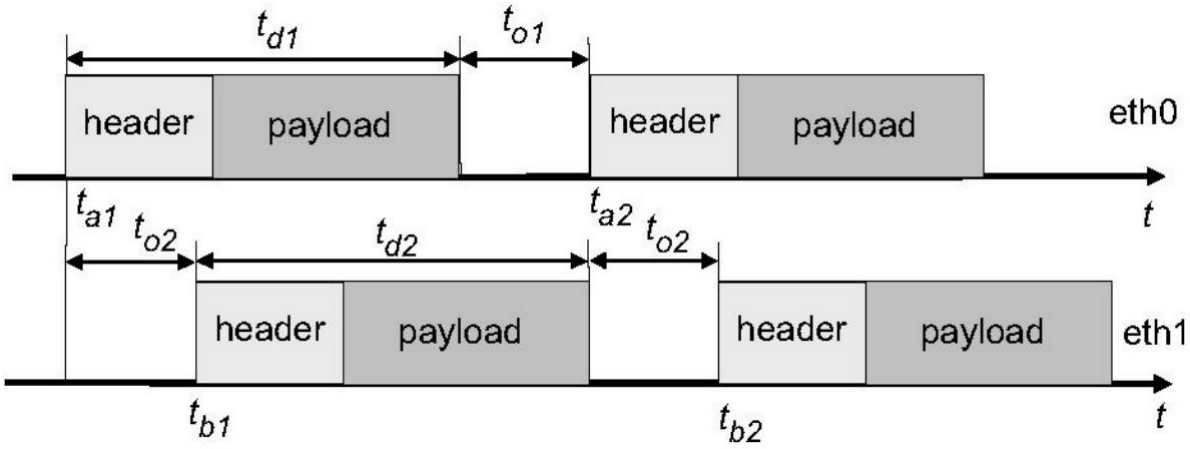
A agregação de *link* no nível de *hardware* permite que o tráfego seja distribuído em paralelo entre portas correspondentes, atuando logicamente como um único *link*, somando a largura de banda de todas as interfaces. Atualmente, permite-se conexões dentre 2 e 16 canais físicos em um único canal virtual para aumentar a largura de banda.

Os dispositivos que compõem o núcleo são interligados entre múltiplos canais, permitindo melhor roteamento e alta disponibilidade. Quando algum *link* redundante falha, os pacotes podem ser enviados para os *links* restantes compostos pela agregação de *link*. Isso, de forma transparente para as demais interfaces da rede.

Cada fabricante desenvolve seus equipamentos para a camada 2, como *Split Multi-Link Trunking* que suporta agregação de *link*. Já para os protocolos de camada 3, como *Link Aggregation Control Protocol (LACP)* e *Virtual Concatenation (VCTA)* podem ser desenvolvidos para agregação de *link* e agregação de portas.

De acordo com *Aust et al. (2006)*, existe um limite de taxa de transferência entre as interfaces ligadas, ocasionado pelo deslocamento ocorrido na ligação. Citando o algoritmo Round-Robin, o *throughput* b_m é referente a duas interfaces de rede exemplificados na Figura 2.3.

Figura 2.3: Temporização de pacotes entre duas interfaces de redes vinculadas.



Fonte: Extraído de *Aust et al. (2006)*

O tamanho do pacote $_p$ é o mesmo para cada *link* e não há diferença na taxa de transferência de ambos os *link*. Nesse caso, o tempo de transmissão t_d para cada pacote é o mesmo e pode ser configurado para t_{d1} = t_{d2}. Pode também assumir-se que o deslocamento de ligação para ambos os links é o mesmo, uma vez que não há diferença observada na taxa de transferência dos *links*. O deslocamento de ligação pode ser definido como t_{d1} = t_{d2}.

$$b_m(t_{d1}, t_{d2}, t_{o1}, p) = \frac{p}{(t_{d1} + to_1)} + \frac{p}{(t_{d2} + to_2)})$$

Teoricamente, o tamanho do pacote, o tempo de transmissão e o deslocamento de ligação conduzem à seguinte equação para calcular a taxa de transferência para interfaces n-*bond* (n> 1)

$$b_m(p, n, t_d, t_0) = p.\frac{n}{(t_d + (n-1).t_o)}$$

O parâmetro n é o número de interfaces é t_d e o tempo de transmissão para um frame *Ethernet*. O *bonding* de ligação pode ser calculado da seguinte forma:

$$b_m(p, n, b, t_d) = \left(\frac{p.n}{b}\right).\frac{1}{(n-1)}$$

2.1.3 MLT - (*Multi-Link Trunking*)

O modelo de agregação *Multi-Link Trunking* fornece uma conexão ponto-a-ponto agregando várias portas com a finalidade de agir logicamente como uma única porta.

Segundo Lapuh e Homma (2008), MLT (*Multi-Link Trunking*) é um método proprietário para utilizar múltiplas conexões físicas entre *switches* ou entre o *switch* de um servidor com várias placas de interface de rede como um único *link* lógico. É uma forma avançada de *trunking* que proporciona flexibilidade aprimorada através da ampliação da largura de banda e de agregação de *links*.

Se um ou mais de um *link* falhar, a tecnologia *MLT* redistribuirá automaticamente o tráfego nos *links* restantes. Essa redistribuição automática é realizada em menos de 0,5 segundos. Portanto, nenhuma interrupção é notada pelos usuários finais. Esta recuperação de alta velocidade é exigida quando a rede fica críticas, sendo necessária a reconexão imediata.

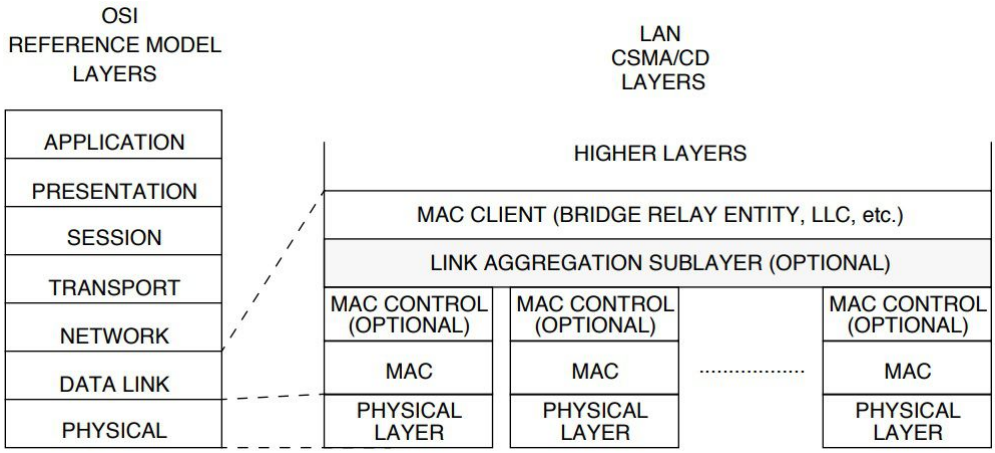
2.1.4 LACP - *Link Aggregation Control Protocol*

Para controlar a agregação de *link*, o LACP permite que um dispositivo de rede controle o agrupamento automático de *links* enviando pacotes LACP para o dispositivo onde está implementando este protocolo.

Segundo Bondan, Gobbi e Fochi (2012), a definição de *LACP* (*Link Aggregation Control Protocol)* consiste em um algoritmo constituído por máquinas de

estados e funções que torna possível através de trocas de pacotes combinando diversos enlaces físicos em um único enlace lógico. Vários benefícios compõem esta técnica, tais como aumento na largura de banda, redundância e balanceamento de carga nos enlaces físicos. Na Figura 2.4 é descrita a disposição do protocolo *LACP* em relação ao modelo de referência *OSI*.

Figura 2.4: Disposição do Protocolo LACP em Relação ao Modelo de Referência OSI.



Fonte: Extraído de Bondan, Gobbi e Fochi (2012)

Este protocolo apresenta algumas limitações que afetam seu funcionamento, como por exemplo; somente pode ser executado em agregações ponto-a-ponto; não suporta *MACs* fora do padrão *IEEE 802.3*; os dados somente trafegam em um sentido no canal; todos os dados devem somente operar na mesma taxa de transferência. A nomenclatura utilizada para os pacotes deste protocolo é *LACPDU* (*Link Aggregation Control Protocol Data Unit*), o qual consiste em informações do sistema de origem do pacote, informações do *host* que ele está enviando o pacote e também informações sobre o sistema de destino.

A funcionalidade do *LACP* restringe-se a *switches* e servidores, onde os dois lados da conexão devem conter suporte ao protocolo. Na Figura 2.4 consta uma exemplificação das camadas de rede em que o protocolo atua diretamente.

Segundo Sarabando (2008), LACP (*Link Aggregation Control Protocol*) faz parte da especificação *IEEE 802.3ad* que permite o agrupamento de várias portas

físicas para formar um único canal lógico. Quando alterado o número de portas agrupadas ativas em um canal, os padrões de tráfego refletem o estado reequilibrado ao *link*. O LACP suporta a criação automática de canais com portas *Gigabit Ethernet*, trocando pacotes entre estas.

Os pacotes do *LACP* são trocados entre portas de modo ativo e passivo. No modo ativo, a porta inicia negociações com portas remotas enviando pacotes. Já no modo passivo, o LACP aloca uma porta a qual responde aos pacotes que recebe, não iniciando a negociação com o *LACP*. Neste modo, o grupo de canal de porta atribui a interface ao pacote. Ambos os modos permitem que os pacotes negociem entre as portas para determinar se podem formar um canal com base em critérios como a velocidade da porta e o estado do *trunking*.

2.1.5 *Multi-Device Link Aggregation*

Segundo Sarabando (2008), o *link* Multi-Dispositivo (MDLA) permite a qualquer dispositivo empregar o padrão *IEEE 802.3ad* para agregação de *link*, comunicado e conectando os dispositivos que usam o mesmo padrão.

Os congestionamentos podem acontecer com os pacotes através do *link*, para corrigir isto, a *IEEE 802.3ad* cria sub-camadas opcionais permitindo que um ou mais *links* sejam agregados. Desta forma, um grupo de agregação é criado com um único *link* lógico, funcionando com várias velocidades, particularmente de 10*Mbits*/segundos a 1000*Mbits*/segundos. Além disso, os usuários podem estabelecer *links Multi-Gigabit*, proporcionando assim maior largura de banda.

2.1.6 VCAT - *Virtual Concatenation*

Segundo Loh (2012), o *VCAT* combina essencialmente a largura de banda útil do sinal da rede em camadas múltiplas para suportar um único *link* de camada para rede.

A concatenação virtual permite extrair toda a largura de banda requerida de uma malha, uma vez que podem seguir caminhos diferentes através da rede para atender a demanda solicitada pelos usuários que à ocupam. O *VCAT* realiza a multiplexação inversa a cada oito *bytes* de intercalação do encapsulamento de *bits* da rede.

Com *LCAS* habilitado, o membro que falhou é temporariamente retirado do serviço que fornece o conjunto do Grupo *VCAT*, até que a falha seja reparada.

O *VCAT* é aplicado nos pontos finais das conexões, o que permite que cada canal seja transmitido de forma independente através de uma rede de transporte herdada.

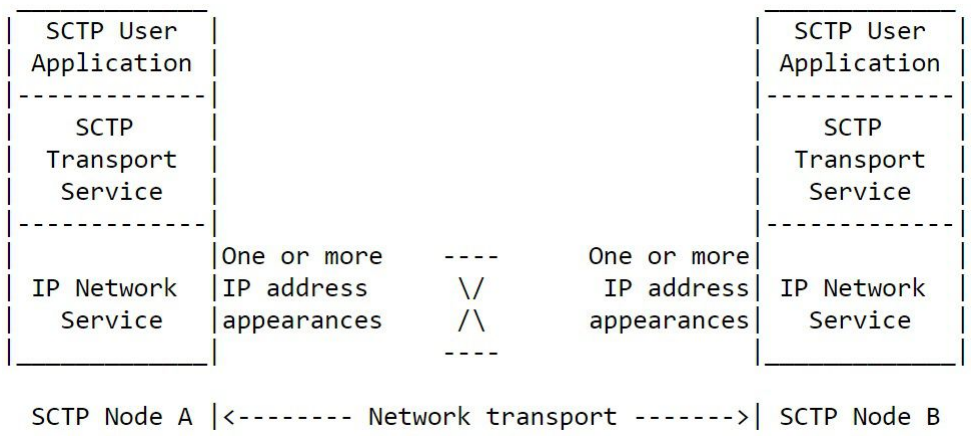
2.1.7 *Stream Control Transmission Protocol* - (SCTP)

Segundo Ong (2002), o *Stream Control Transmission Protocol* (SCTP) é um protocolo da camada de transporte, estando no mesmo nível de UDP (*User Datagram Protocol*) e TCP (*Transmission Control Protocol*). Este protocolo fornece um serviço de transporte confiável, onde os dados são transportados através da rede sem erros e em sequência.

O SCTP possui um mecanismo orientado a associação, o que significa que é criado um laço de relacionamento entre os pontos finais de SCTP antes da transmissão dos dados. Esta relação é mantida até que a transmissão seja concluída com sucesso. A Figura 2.5 demostra este modelo de associação.

O SCTP tem por característica ser um protocolo unicast suportando a troca de dados entre exatamente 2 pontos finais, embora estes possam ser representados por múltiplos endereços IP. Fornecendo uma transmissão confiável, o protocolo detecta os pacotes descartados, reordenados, duplicados ou corrompidos e retransmite estes dados danificados conforme necessário. A transmissão SCTP é *full duplex*. Por ser orientado a mensagem, o SCTP suporta o enquadramento dos limites das mensagens individuais.

Figura 2.5: Modelo de Associação.



Fonte: Adaptado de Ong (2002)

Conforme o Shi et al. (2003), uma particularidade do SCTP é transferir dados através de vários fluxos. Outra característica é a utilização da redundância a falhas. As características citadas acima são responsáveis por trazer mais eficiência, robustez e segurança ao protocolo.

2.1.8 *Multichannel* CSMA

Segundo Jain (2001), no *Multichannel CSMA Carrier Sense Multiple Access*as utilizações de múltiplos canais oferecem inúmeras vantagens em relação ao desempenho na redução de colisões. Isso possibilita transmissões simultâneas, melhorando o uso da largura de banda mesmo utilizando-se de agregações. Os protocolos multicanal permitem a utilização de nós próximos transmitindo simultaneamente os pacotes em diferentes canais sem interferência entre si.

Como exemplificação, a largura de banda (B) pode ser dividida em N canais FDMA *(Frequency Division Multiple Access)* de largura menor, cada um com uma largura de banda B/N. Esta mesma teoria é aplicada ao protocolo *(Code Division Multiple Access)*. Os resultados do uso de MAC baseados em CSMA garante com que o tráfego é distribuído entre vários canais N, assim reduzindo a disputa por canais individuais. Em contrapartida, o efeito indesejado é o aumento do tempo de transmissão de pacotes em N vezes. De acordo com a forma de utilização, os

protocolos multicanais MAC podem ser classificados em dois, na qual a primeira classe assume um canal separado para cada nó na rede.

A primeira classe pode ser orientada ao transmissor, onde cada nó transmite utilizando seu próprio código, ou orientada ao receptor, onde todas as transmissões são feitas no mesmo receptor usando o mesmo código. A segunda classe de protocolos multicanal MAC não assume canais dedicados para cada nó, a largura de banda disponível é supostamente dividida em uma série de canais.

2.1.9 MPTCP - (*Multi-Path TCP*)

O Protocolo de Controle de Transmissão (TCP) é usado pela grande maioria dos aplicativos para transportar seus dados de forma confiável pela Internet. O TCP é a comunicação comum, os dispositivos móveis ou computadores com muitas interfaces de rede TCP sabiam que os *links* de rede podiam falhar, e ocorrer a perda de dados ou demorar na entrega.

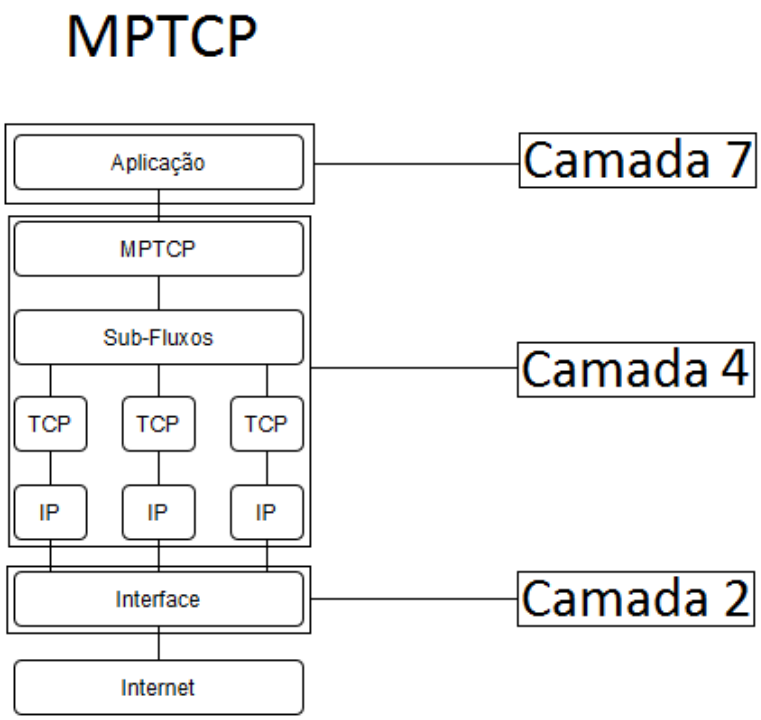
As redes atuais são *multipath*, onde os dispositivos móveis têm múltiplas interfaces sem fio, os *data centers* têm muitos caminhos redundantes entre servidores, e o *multihoming* tornou-se a norma para grandes grupos de servidores. Enquanto isso, o TCP é essencialmente um protocolo de caminho único: quando uma conexão TCP é estabelecida, a conexão é vinculada aos endereços IP dos dois *hosts* comunicantes. Se um desses endereços mudar, por qualquer motivo, a conexão falhará.

Segundo Scharf e Ford (2006), o MPTCP tem como finalidade explorar caminhos de comunicação disponíveis, são utilizados múltiplos caminhos para troca de dados. *Multipath* TCP é uma extensão do protocolo TCP padronizada pela IETF. Como tentativa de resolver grandes falhas do TCP. Na maneira de como usar dispositivos para se comunicar, foram desenvolvidas um novo protocolo, como o MPTCP. Com o surgimento de novos recursos, os dispositivos móveis podem ter acesso ilimitado à comunicação. Todos os dispositivos estão se tornando *multipath*,

contemplando várias conexões.

Sabe-se que o TCP é um caminho único, isso significa que pode haver apenas um caminho entre dois dispositivos que tenham a sessão TCP aberta. Esse caminho é selado como uma sessão de comunicação definida pelo endereço IP de origem e destino. Na Figura 2.6 é demostrado um exemplo da arquitetura do protocolo MPTCP.

Figura 2.6: Exemplo de Arquitetura MPTCP.



Fonte: Adaptado de Sarabando (2017b)

Se algum dispositivo mudar a comunicação de 3G para *Wireless*, a sessão TCP é desconectada e uma nova comunicação é criada através do Wi-Fi. Basicamente, quando o MPTCP iniciar a comunicação, vários caminhos na rede são criados. Quando isso acontece, o TCP que está trabalhando é bloqueado, deixando o MPTCP definir o controle da rede por sub-sessões.

Multipath TCP trabalha na camada 4, criando subfluxos pelos quais são transmitidos os dados, e desta forma, obtendo melhor desempenho se comparado ao TCP, distribuindo a carga do trafego para mais de um subfluxo. O MPTCP é a

evolução do TCP padrão que faz o transporte de pacotes de dados *multipath* em possível conexão.

Segundo Popeskic (2014), para tornar a transmissão multicanal possível, são necessárias várias sequências de dados. No TCP normal, o número de sequência de dados é usado para colocar os segmentos em ordem quando todos os pacotes chegarem ao receptor. Porém, os pacotes podem chegar ao receptor usando mais de um sub-fluxo dentro de uma conexão MPTCP, e quando chegar ao receptor, o mesmo junta-se deixando exatamente como saiu da origem.

Para abrir uma conexão normal, o cliente envia um pacote SYN para sincronizar à porta na qual o servidor está escutando. O pacote SYN contém a porta de origem e o número de sequência inicial escolhido pelo cliente e pode conter opções de TCP usadas para negociar o uso de extensões. O servidor responde com um pacote SYN + ACK, reconhecendo o SYN e fornecendo o número de sequencia inicial do servidor e as opções que ele suporta. O cliente reconhece o SYN + ACK, e a conexão está agora totalmente estabelecida.

Como os dois caminhos terão frequentemente diferentes características de atraso, os segmentos de dados enviados nos dois sub fluxos não serão recebidos por ordem. O TCP regular usa o número de sequência em cada cabeçalho do pacote TCP para colocar os dados de volta na ordem original. Alguns descartam os segmentos fora de sequência, enquanto outros tentam atualizar os reconhecimentos TCP para recuperar algumas lacunas.

Muitos caminhos estão disponíveis entre dois pontos finais, e o roteamento *Multipath* escolhe aleatoriamente um para uma conexão TCP específica. Isso pode causar colisões, onde vários fluxos são colocados no mesmo *link* prejudicando o *throughput*, assim reduzindo a eficiência da comunicação.

Multipath TCP também pode ajudar em outros cenários. Por exemplo, os servidores *Web Multihomed* podem executar balanceamento de carga em suas ligações, deixando que os *Hosts* de duas pilhas podem usar a comunicação do

IPv4 e o IPv6 dentro de uma única conexão, portanto, fazendo a comunicação entre elas mesmo sendo de protocolos diferentes.

2.1.9.1 Equal Cost Multipath - ECMP

Segundo Sousa, Moreira e Machado (2009) a ideia básica é garantir que exista uma rede tolerante a falhas, uma maneira sugerida de melhorar o processo de reencaminhamento era ajustar os temporizadores do protocolo de roteamento.

Isso causou um requisito necessário em conceitos para acelerar os desempenhos de recuperação das redes de dados. Foi o que levou a existência do recurso ECMP. A ECMP carrega fluxos em vários caminhos de IP. Esse recurso, disponível para vários protocolos de roteamento, permitiu uma distribuição de pacotes em vários *links* de saída entre a origem e o destino.

O *Multipath* de custo equivalente roteia o tráfego ao longo de todos os caminhos existentes de custo igual. Cada roteador mantém uma tabela de roteamento de todos os caminhos de custo igual para destinos em uma rede e encaminha os pacotes para um determinado destino equilibrado nos *links* apropriados. Além da resiliência contra falhas de nó e *link*, o ECMP fornece uma distribuição de tráfego mais equilibrada na rede e um maior débito para os padrões típicos.

O algoritmo ECMP pode ser definido para equilibrar pacotes ou fluxos. No primeiro caso, cada roteador determina o próximo salto para um pacote com base em seu destino. No segundo caso, o caminho é determinado da mesma maneira para o primeiro pacote de um fluxo, enquanto os pacotes subsequentes do fluxo seguem o mesmo caminho.

A vantagem do balanceamento de fluxo é que os pacotes são entregues em ordem. No entanto, o equilíbrio é imperfeito, pois os fluxos têm tamanhos diferentes. Para um equilíbrio mais uniforme, o monitoramento de fluxo e a atribuição aos caminhos devem ser implementados, o que aumenta a complexidade do roteamento.

O ECMP também é usado para redirecionamento rápido de IP, ou seja, para o pre-roteamento de segundos em redes IP. Se um roteador detectar a falha de um *link*, pode redistribuir rapidamente o tráfego para outro caminho disponível no roteador, no entanto, esta solução abrange um grande conjunto de cenários para a soluções à falha, uma vez que tenha um caminho alternativo.

2.1.9.2 Internet Control Message Protocol (ICMP)

Gont (2010) diz que o ICMP é usado na arquitetura da Internet principalmente para executar a função de isolação de falhas, ou seja, o problema que acontece é isolado, assim nao prejudicando outra função da rede.

Quando um roteador intermediário detecta um problema de rede, ao tentar encaminhar um pacote IP, geralmente enviará uma mensagem de erro ICMP para o sistema fonte, para informar sobre o problema de rede que está ocorrendo. As mensagens de erro ICMP são transmitidas sem problemas e podem ser descartadas devido a corrupção de dados, congestionamento de rede ou limitação de taxa.

Este protocolo não corrige o erro, mas mostra quais são as possíveis camadas podem resolver o erro que está na rede. Vários problemas o ICMP pode identificar, sendo erros de sistema, de processo ou mesmo de rede. Os erros são corrigidos momentaneamente, evitando com que o problema se agrave.

2.1.9.3 Energy Efficient Multipath Transmission Control Protocol (EMPTCP)

O eMPTCP usa uma combinação de gerenciamento de sub-fluxo, o gerenciamento de sub-fluxo permite que o eMPTCP escolha os caminhos eficientes para ter a eficiência energética *per-byte*, usando medidas de tempo de execução e um modelo de consumo de energia permanente.

O objetivo do eMPTCP é reduzir o consumo de energia em relação ao

MPTCP, minimizando o atraso adicional. Este protocolo faz isso, escolhendo dinamicamente caminhos com base na eficiência energética estimada.

A aplicação do eMPCTP requer um modelo de energia que capture o consumo de energia de acordo com o uso da interface de rede. O uso de eMPTCP em dispositivos móveis mostraram que é capaz de consumir menos energia comparando com o MPTCP, enquanto fornece benefícios do MPTCP de múltiplos caminhos, melhor desempenho, disponibilidade e transparência. A única diferença do eMPTCP em relação ao MPTCP, é a economia de energia.

2.1.10 SDN - *Software Defined Network*

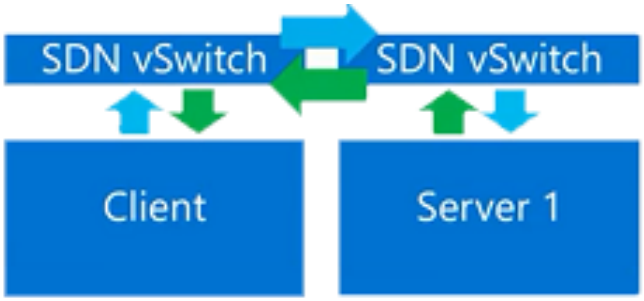
O SDN (*Sofware Defined Networking*) é um termo abrangente que engloba vários tipos de tecnologia de rede com o objetivo de tornar a rede ágil e flexível com servidor virtualizado.

De acordo com Andrioli, Rosa Righi e Aubin (2017), o objetivo de tornar uma rede ágil flexível o SDN consiste em um *software* administrador de rede com o poder de moldar o tráfego de controle, centralizando sem precisar tocar em *switchs*.

Segundo McIllece e Poggemeyer (2017), o SDN se concentrou exclusivamente na separação do plano de controle da rede, que toma decisões sobre como os pacotes devem fluir através da rede. Quando um pacote chega a um interruptor na rede, as regras incorporadas no *firmware* proprietário indicam ao *switch* onde encaminhar o pacote para o destino final.

O *switch* envia todos os pacotes para o mesmo destino ao longo do mesmo caminho e trata todos os pacotes exatamente da mesma maneira. Em um cenário SDN, as regras para o processamento de pacotes são enviadas para o *switch* de um controlador, no qual um aplicativo executa a consulta no controlador para orientação conforme necessário, fornecendo informações sobre o tráfego que estão manipulando. Controladores e *switches* se comunicam através da interface do controlador, geralmente *OpenFlow*.

Figura 2.7: Arquitetura SDN.



Fonte: Adaptado de Microsoft (2014)

Quando uma rede tradicional usa um dispositivo especializado, como um *firewall* ou um SDN, implementa-se um aplicativo que usa o controlador para gerenciar o comportamento do plano de dados. A rede definida por *software* usa um modo de operação chamado de adaptativo ou dinâmico no qual um *switch* emite uma solicitação de rotas para um controlador para um pacote que não possui uma rota específica. Esse processo é separado do roteamento adaptativo, que emite solicitações de rotas através de roteadores e algoritmos com base na topologia da rede, e não através de um controlador.

Com o SDN, o administrador pode alterar as regras de qualquer rede quando necessário, despropriando ou mesmo bloqueando tipos específicos de pacotes com um nível muito grande de controle. Isso é especialmente útil em uma arquitetura *multi-tenant* de computação em nuvem, pois permite que o administrador gerencie as cargas de tráfego de forma flexível e eficiente. Essencialmente, isso permite que o administrador use regras de *commodities* menos forçadas e tenha mais controle sobre o fluxo de tráfego da rede.

O SDN tem por característica proporcionar uma abordagem mais dinâmica, gerenciável e adaptável ao gerenciamento de redes, lidando com a largura de banda em aplicações atuais tornando as redes mais eficientes e flexíveis. A programação é feita diretamente pelos administradores, que serão responsáveis pela configuração, gerenciamento, proteção e otimização dos recursos de rede com agilidade.

O ponto positivo da programação da rede pelos administradores é prover o ajuste do fluxo de tráfego em toda a rede, ajustando conforme as demandas de mudança, assim toda a inteligência da rede é centralizada em *software*. O plano de controle é responsável por atender o roteamento dos pacotes, determinando qual rota seguir, assim construindo uma tabela de roteamento, já o plano de dados roteia os pacotes baseados em regras simples, baseada em cada entrada feita na tabela de roteamento do dispositivo.

Levando em consideração o desempenho da rede, os controladores *SDN* respondem aos pedidos dos dispositivos de rede e conseguem reconfigurá-lo com agilidade. Seu diferencial das demais arquiteturas está na adaptação ao crescimento da rede com transparência. Segundo os estudos realizados por Andrioli, Rosa Righi e Aubin (2017), o consumo energético aumenta em torno de 11 % para cada *link* virtual adicionado, sendo estes *links* compostos por um *switch* de 8 portas.

Uma tecnologia SDN proposta para padronizar a forma como um controlador se comunica com dispositivos de rede em uma arquitetura SDN. O *OpenFlow* fornece uma especificação para migrar a lógica de controle de um *switch* para o controlador. Ele também define um protocolo para a comunicação entre o controlador e os *switches*.

As arquiteturas baseadas em *OpenFlow* possuem capacidades específicas, ideias de testar novas aplicações. Esses recursos incluem análise de tráfego baseada em software, controle centralizado, atualização dinâmica de regras de encaminhamento e captação de fluxo. Os aplicativos baseados em *OpenFlow* foram propostos para facilitar a configuração de uma rede, simplificar o gerenciamento de rede e adicionar recursos de segurança, virtualizar redes e data centers e implantar sistemas móveis.

O *OpenFlow* é uma tecnologia emergente que torna os elementos de rede, como roteadores ou *switches*, programáveis através de uma interface padronizada. Ao usar a virtualização e o roteamento baseado em fluxo, o *OpenFlow* permite

uma rápida implantação de novos algoritmos de encaminhamento e roteamento de pacotes, focando em redes fixas. Para recuperação rápida de falhas, o controlador deve notificar todos os *switches* afetados sobre a ação de recuperação dentro do intervalo de *megabits*/s.

2.2 COMPUTAÇÃO EM NUVEM

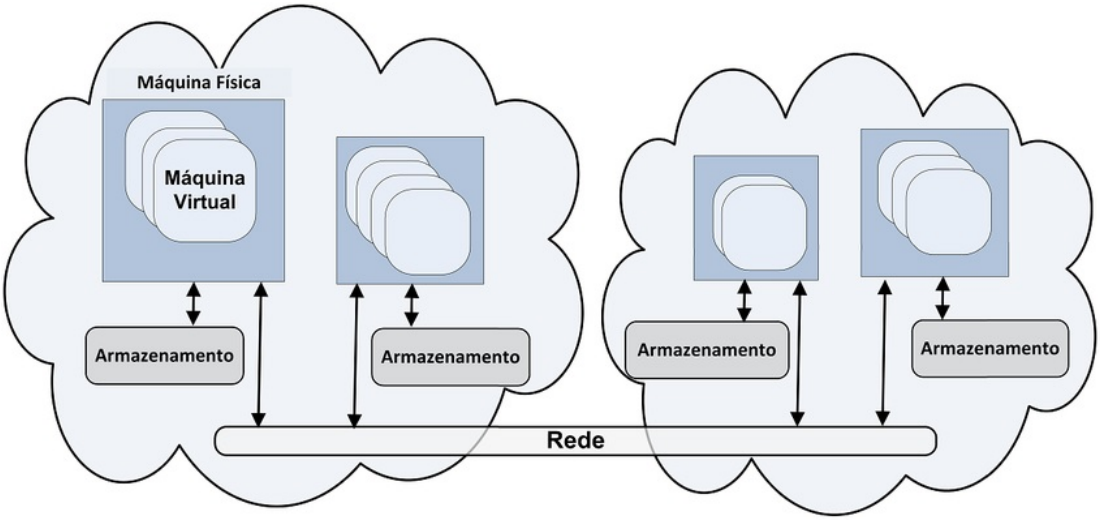
A principal característica da computação em nuvem consiste em oferecer recursos de TI através da internet sendo entregue sob demanda. Com a finalidade de padronizar esta tecnologia, a ISO/IEC17789 (2014) estabelece um conjunto de regras que devem ser seguidas para a eficiente implementação.

A computação em nuvem divide-se em dois segmentos, os modelos de serviços e os modelos de implementação. Para caracterizar os modelos de serviços, define-se um padrão de arquitetura, subdivididos em três modelos, SaaS, PaaS e Iaas. Caracterizando os modelos de implementação, a literatura descreve como um meio de restringir o acesso e disponibilidade dos ambientes de computação em nuvem, que são a Pública, Privada, Comunitária e Hibrida.

Segundo Sousa et al. (2010), a computação em nuvem está se tornando uma das mais crescentes tecnologias na indústria da TI, por ocultar a complexidade total da infraestrutura facilitando a implementação de centro de dados com a utilização de *hardware* compartilhado de armazenamento. Normalmente a infraestrutura do ambiente de computação em nuvem é composto por centenas e milhares de dispositivos ou nós físicos de baixo custo conectada pela rede como demostra a Figura 2.8.

Existem variações de *hardware* como CPU, memória, armazenamento em disco, que varia de acordo com a necessidade de cada *Data Center*, porém as configurações de *software* são as mesmas em todas as máquinas. Cada máquina possui um número variável de máquinas virtuais em execução, conforme a disponibilidade de *hardware* que cada dispositivo suporta.

Figura 2.8: Computação em Nuvem.



Fonte: Extraído de Coutinho et al. (2013)

Quer seja chamado de computação em nuvem ou computação *on-demand*, software como serviço ou a *Internet* como plataforma, o elemento comum é uma mudança na geografia da computação. Quando cria-se uma planilha com o serviço *Google Docs*, os principais componentes do software residem em computadores invisíveis, paradeiro desconhecido, possivelmente espalhado por continentes. Um computador cliente na *Internet* pode se comunicar com vários servidores ao mesmo tempo, alguns dos quais também podem trocar informações entre si.

O objetivo da computação em nuvem é concentrar o armazenamento no núcleo, onde as máquinas de alto desempenho estão ligadas por conexões de alta largura de banda, e todos esses recursos são cuidadosamente gerenciados. Na computação em nuvem é definido como evolução da tecnologia sobre demanda, que tem por objetivo prover componentes básicos como armazenamento, processamento e largura de banda através de provedores especializados oferecendo este serviço com baixo custo e larga flexibilidade.

A preocupação que o usuário terá será somente a forma de acesso aos serviços em seu *link* local, pois o provedor do serviço em nuvem dispõe de disponibilidade total a qualquer tempo, sem bloqueios, sem recursos limitados pelos parâmetros do sistema. O que tornam os serviços de computação em nuvem aces-

síveis é o fato do usuário pagar somente por aquilo que está utilizando do poder computacional, não havendo custos iniciais de implantação de infraestrutura e serviços de TI.

Para a computação em nuvem, os serviços dispõe de nuvem privada, nuvem híbrida e nuvem pública. Na computação em nuvem privada, as implantações dos serviços são feitas internos na empresa, evitando a dependência de servidores e provedores externos, e um dos principais problemas a nuvem privada evita os atrasos de troca de informações e dados. As principais características da computação em nuvem vêm das vantagens em soluções como *Self-service* sob demanda onde o usuário adquire recursos computacionais conforme sua necessidade, sem a necessidade de interação humana para os provedores de cada serviço. A elasticidade rápida é um dos fatores que dá ao usuário recursos quase que ilimitados, de forma rápida e elástica, quase que automaticamente em alguns casos.

2.2.1 Modelos de Serviço

Segundo Medeiros (2015), afim de definir um padrão de arquitetura para computação em nuvem, é formada uma descrição composta por três modelos de serviços. Começando pelo *SaaS* (*Software* como um Serviço), *PaaS* (Plataforma como um Serviço) e *IaaS* (Infraestrutura como Serviço) demonstrado na Figura 2.9.

Figura 2.9: Serviços de Nuvem

Fonte: Extraído de Microsoft Azure (2017)

2.2.1.1 Software como um Serviço (SaaS)

O *SaaS* é definido por um sistema de *software* com propósitos específicos, os quais são disponíveis aos usuários pela *Internet*. O usuário acessa o serviço por meio de uma interface *thin client* utilizando um navegador *Web*, porém o usuário não tem permissão de controle sobre os servidores, armazenamento, características individuais da aplicação e sistemas operacionais. Pelo fato do *software* estar disposto na *Internet*, seu acesso se torna facilitado por não haver restrição em relação a localização do usuário e disponibilidade do serviço, reduzindo custos por não ser necessária a aquisição de licenças de uso.

Segundo Kavis (2014), essa tecnologia foi criada para facilitar e agilizar o dia a dia do cliente, onde toda a infra-estrutura estará implantada em *data centers*, e o cliente não vai precisar de nenhuma infraestrutura de TI avançada. Os usuário possuem diretório dos dados e ferramentas que utilizam, mas não necessitam de aquisição de licenças ou comprar os softwares.

Um dos exemplos, desta tecnologia é o *Google* que utiliza o *Google Docs*, permitindo o acessos às ferramentas pela Internet, não precisando fazer a instalação de nenhum aplicativo, esse é um dos exemplos da aplicação *SaaS*.

2.2.1.2 Plataforma como um Serviço (PaaS)

De acordo com Kavis (2014) diferentemente do *Software as a service*, o *Platform as a Service* fornece infraestrutura de alto nível de integração ao usuário. Podendo assim testar e implementar aplicações em nuvem, mas não fornece acesso para controlar a infraestrutura de rede, como servidores, armazenamento e as configurações implantadas nesta infraestrutura.

Por conter ferramentas de colaboração entre desenvolvedores, o PaaS dispõe de linguagens de programação, sistema operacional e também ambientes de desenvolvimento para *software*. Seu ambiente de desenvolvimento é escalável,

mas em contrapartida devem ser respeitadas algumas limitações impostas pelo ambiente, o que se diz através da concepção ás aplicações, aplicações em sistema de gerenciamento de bando de dados.

2.2.1.3 Infraestrutura como um Serviço (IaaS)

Com a função de prover toda infraestrutura necessária ao PaaS e o SaaS, o IaaS facilita e torna acessível o fornecimento dos recursos a estes serviços, como servidores, rede, armazenamento e todas as funções para construção do ambiente sob demanda. A API (*Application Programming Interface*) fornece uma única interface assim facilitando a administração da infraestrutura, estabelecendo a comunicação entre *hosts*, roteadores, *switches*, balanceadores de rede.

Geralmente o usuário não administra a infraestrutura em nuvem, mas controla os sistemas operacionais, implantação de aplicativos, armazenamento. A definição de IaaS consiste em técnicas para virtualizar recursos de computação baseadas na infraestrutura computacional, escalando dinamicamente de acordo com a demanda exigida pelas aplicações. A economia trazida por este recurso se faz através do aproveitamento dos equipamentos de rede e servidores ampliando os serviços através dos recursos disponíveis adicionando outros servidores virtuais à infraestrutura já existente dinamicamente.

2.2.2 Modelos de Implementação em Nuvem

Surgido a necessidade de gerar restrições aos usuários, 4 modelos de implementação em nuvem foram desenvolvidos para suprir a demanda de algumas empresas que desejam privar usuários de ambientes mais restritos. Segundo Medeiros (2015), existem quatro tipos de modelos de implementação em nuvem: Pública, Privada, Híbrida e Comunitária, que podem depender de serviços terceirizados ou mantidos por algum provedor de serviço privado.

Uma característica do modelo Público é oferecer serviço para empresas

públicas ou de grande porte com grande poder de armazenamento e processamento provendo acesso rápido aos recursos de computação de forma financeiramente suportável para esta Instituição.

Assim, a compra de *hardware*, *software* e infraestrutura fica por conta do provedor do serviço e não por conta do usuário. Para este modelo de implantação, a disponibilidade da infraestrutura de nuvem é pública, onde quaisquer usuários podem acessar, desde que tenha conhecimento da localização do serviço.

Para o modelo de aplicação público não há restrições de acesso quando se refere ao gerenciamento de redes, autenticação e autorização. Os principais benefícios trazidos pelo modelo de computação em nuvem público começa pelo acesso rápido aos recursos de TI, redução dos custos de infraestrutura, acesso descentralizado.

O modelo de nuvem privada tem como característica a utilização de sua infra-estrutura por uma única organização sendo ela a proprietária onde é taxado um aluguel pela exclusividade da mesma.

Conforme Medeiros (2015), a nuvem privada muitas vezes deixa a cargo do cliente ou organização o gerenciamento da infraestrutura, assim facilita a utilização por empresas e não somente pelo uso público. Pelo fato da empresa ou Instituição gerenciar o processamento e armazenamento, a Instituição ou empresa pode implantar políticas de acesso aos serviços, tais como gerenciamento de redes, utilização de modelos de autenticação ou autorização e configurações dos provedores de serviços.

Os benefícios trazidos pela adoção da nuvem privada são a agilidade na implementação e tempo de implantação dos serviços de TI, acesso aos aplicativos e informações em qualquer lugar.

A nuvem híbrida mescla dois modelos de serviço, a nuvem privada e a nuvem pública. Utilizando-se de alguns recursos da nuvem privada como *hard-*

ware e *software* e outros recursos da nuvem pública como terceirização de alguns serviços, a nuvem híbrida atua como uma única entidade, conectada através de tecnologias proprietárias ou padronizadas possibilitando a acessibilidade de dados e aplicações.

O público que mais adere a este modelo de nuvem computacional são entidades governamentais e corporativas de grande e médio porte, aperfeiçoando e eficiência e agilidade dos negócios. utros benefícios da nuvem híbrida estão na flexibilidade, economia de recursos, proteção aos ativos da informação.

A modelo de implantação de nuvem comunitária não é muito utilizada por ser um modelo colaborativo, onde duas ou mais instituições com o mesmo objetivo em comum se unem para alcançar os mesmos resultados, podendo ser administrada por alguma das organizações ou por terceiros. Para este modelo há a possibilidade de monitorar os recursos cobrando os serviços de acordo com um contrato taxados conforme a demanda e somente os membros com o mesmo interesse tem acesso.

2.2.3 Plataformas de Nuvem

As plataformas em nuvem são meios de aplicações de infraestrutura de alto nível, para implementar, desenvolver e testar na nuvem. Cada aplicação desenvolvida em nuvem, oferece diversas linguagens e sistemas operacionais facilitando a agregação de *softwares* e colaborando no desenvolvimento de aplicações.

2.2.3.1 OpenStack

Segundo Wen et al. (2012), o *OpenStack Compute Service* pode ser usado para hospedar e gerenciar sistemas de computação em nuvem, implementando recurso para acessos virtuais com auto-atendimentos a bibliotecas.

O *OpenStack* nasceu de um projeto da *Rackspace* e da *NASA*, é usado

para construir nuvens *IaaS* públicas e privadas, que trabalha com um conjunto de componentes para implantar redes virtuais e discos virtuais, sendo composto por *software* livre e código aberto, desenvolvida em *Python* e com interface *WEB* para facilitar a integração com o usuário.

Um dos principais pontos de destaque do *OpenStack* é a possibilidade de criar ferramentas para gerenciamento e construção de nuvem privada e pública, liberando o uso das capacidades conforme a necessidade de cada usuário.

De acordo com OpenStack (2011), *Glance* é dividido em camadas e cada uma das camadas tem sua tarefa específica, usando um banco de dados que é compartilhado entre todos os componentes no sistema e é baseado em *SQL* por padrão. Outros tipos de *backends* de banco de dados são suportados e usados pelos operadores,para as bibliotecas compostas pelo *OpenStack Glance* são responsáveis pela interação de armazenamentos externos e de armazenamentos de sistemas locais.

Conforme Red Hat (2017), o *Cinder* é um serviço de armazenamento em bloco para o *OpenStack*, ele foi projetado para apresentar recursos de armazenamento para usuários finais. O conceito de *Cinder* é virtualizar o gerenciamento de dispositivos de armazenamento em bloco e fornecer aos usuários finais uma *API* de auto atendimento para solicitar e consumir esses recursos sem exigir qualquer conhecimento de onde seu armazenamento está realmente implantado ou em que tipo de dispositivo.

O sistema de armazenamento em bloco gerencia a criação, a conexão e a separação dos dispositivos de bloco em servidores. Os volumes de armazenamento em bloco são totalmente integrados ao *OpenStack Compute* permitindo que os usuários da nuvem gerenciem suas próprias necessidades de armazenamento. Além do armazenamento local do servidor Linux, o usuário pode usar plataformas de armazenamento, incluindo *Ceph, CloudByte, Coraid, EMC*.

Segundo Red Hat (2017), o *OpenStack Identity (Keystone)* fornece um di-

retório central de usuários mapeados para os serviços *OpenStack* aos quais eles podem acessar. Atua como um sistema de autenticação comum em todo o sistema operacional da nuvem. Sua plataforma suporta múltiplas formas de autenticação, incluindo credenciais padrão de nome de usuário e senha, além disso ele tem a escolha dos recursos de disponibilidade que deseja configurar para acessar.

2.2.3.2 OpenNebula

Conforme Soto (2011), *OpenNebula* é conhecida por ser uma ferramenta flexível de *data center* para implantação de redes *IaaS* em nuvem, utiliza ferramentas de virtualização para gerenciar estruturas de nuvem privadas. Sua analogia concentra um grupo de máquinas virtuais, trabalhando no mesmo armazenamento de rede, mesma virtualização, permitindo segurança com os serviços do mesmo local, e combinando recurso como de nuvem remota. Esta ferramenta caracteriza-se por ter um padrão de código aberto para gerenciar a complexidade e infraestruturas de *data centers* distribuídos.

Segundo Vogel et al. (2016), *OpenNebula* surgiu como ferramenta para ambientes de nuvem pública, privada e híbrida, mas tem um recurso principal que é o *front-end* responsável por executar serviços da ferramenta. *OpenNebula* tem por característica a simplicidade à sua gerência, pois não necessita de muitos componentes que podem ocorrer perdas de flexibilidade. Para a implantação da ferramenta é necessário possíveis ajuda de terceiro para efetuar determinadas tarefas desejadas.

Um mecanismo de infraestrutura virtual que permite a implantação dinâmica e a reatribuição de máquinas virtuais em um conjunto de recursos físicos. Estende os benefícios das plataformas de virtualização de um único recurso físico para um conjunto de recursos, desacoplando o servidor da infraestrutura física e da localização física.

De acordo com as políticas de alocação, organiza tecnologias de armaze-

namento, rede, virtualização, monitoramento e segurança para permitir a colocação dinâmica de serviços de várias camadas (grupos de máquinas virtuais interconectadas) em infraestruturas distribuídas, combinando ambos os *data centers* recursos e recursos da nuvem remota.

2.2.3.3 CloudStack

A plataforma *CloudStack* é definida como um *software* de código aberto com recursos computacionais específicos para a construção de nuvens *IaaS*, provendo gerenciamento da rede, *storage* e todos os recursos computacionais que a infraestrutura de nuvem necessita.

De acordo com Roveda et al. (2015), um recurso interessante do CloudStack e que o gerenciamento dos recursos da ferramenta e bem granular. A plataforma *CloudStack* suporta várias formas de virtualização, como *XenServer*, *VMware*, *OracleVM* e *KVM* por exemplo. As principais características da arquitetura desta plataforma contemplam a escalabilidade e balanceamento de carga.

A utilização de *CloudStack* abrange a organização do desenvolvimento de testes de maneira íntegra, facilitando o desenvolvimento bem como processos de publicação de aplicativos.

A arquitetura de implantação do *CloudStack* consiste na instalação de um servidor de gerenciamento de recursos da nuvem e infraestrutura de nuvem, oferecendo então recursos como armazenamento, *hosts* e endereços IP no servidor de gerenciamento.

2.2.3.4 Azure

Segundo Chappell (2010), *Azure* é um sistema operacional *Windows*, baseado em ambiente de desenvolvimento em nuvem, possuindo um plataforma flexível que suporta hospedagem de serviços, gerenciamento de aplicações *Web* na

Internet em *data centers*. Sua plataforma suporta várias linguagens, seus serviços utilizam a plataforma *PaaS* (plataforma de serviços) para executar suas aplicações.

Conforme Huang et al. (2012), esta aplicação é dividida em pilares e cada pilar é responsável por um ambiente, nos quais podem ser controlados separadamente sendo diagnosticados em processamentos, escalabilidade e custos.

2.2.3.5 Amazon

Segundo Azevedo et al. (2012), uma das maiores empresas provedoras de serviços de Nuvem, a *Amazon*, oferece infraestrutura de serviços *IaaS*. O grau satisfatório entre seus clientes, vêm dos da infraestrutura em seus *data centers* distribuídos geograficamente pelo mundo, e serviços beneficentes que valorizam o cliente.

Os servidores da Amazon tem diversas dinâmicas com os clientes, permitindo aumentar ou diminuir suas utilidades em nuvem, adequando às necessidades de criação e remoção de máquinas virtuais e entre outras aplicações que o usuário depende da nuvem.

2.3 VIRTUALIZAÇÃO

A principal importância dos modelos de virtualização se traz pela ampla diversidade de implantação de plataforma de *software* sem a necessidade de se ter um aumento no número de *hardware*. Aderindo algumas características de virtualização, o modelo de virtualização LXC gerencia o particionamento de diversos recursos em um único sistema operacional, porém, sem nenhuma virtualização.

O KVM basicamente é um *kernel* a parte carregando no Sistema Operacional Linux, o qual permite simular diversos sistema operacionais, com o diferencial de ser executado abaixo da camada de virtualização.

Segundo Santos (2011), a virtualização de servidores teve início nas pes-

quisas realizadas pela IBM (*International Business Machines*). Para chegar ao padrão atual vêm se desenvolvendo desde a década de 90 várias ferramentas, mas o amadurecimento da tecnologia se deu recentemente. Com o desenvolvimento de *softwares* de gerenciamento de *hardware* oferecidos pela *VMware*, a virtualização ganhou popularidade.

Voltando a virtualização, seu funcionamento trabalha em cima da multiplexação do *hardware* real, assim viabilizando a criação de várias máquinas virtuais podendo ou não ser uma cópia fiel isolada dentro do *hardware* nativo. O conceito de virtualização baseia-se na aplicação de uma camada de abstração entre as aplicações e o *hardware*.

Isto pode ser feito em nível de *hardware*, onde a camada de virtualização é fixada diretamente na máquina física e apresenta as camadas superiores como no ambiente real. Isto faz com que haja maior compatibilidade de *software*.

Já em nível de sistema operacional ocorre uma partição lógica desenvolvida em uma plataforma fazendo com que cada partição será vista como uma máquina isolada, porém compartilhando o mesmo sistema operacional. Para este caso a virtualização é fixada entre o sistema operacional e as aplicações.

Para o nível de linguagens de programação, a virtualização é um programa de aplicação do sistema operacional, onde uma máquina executa aplicações desenvolvidas para uma linguagem de programação de alto nível.

Os benefícios trazidos pela virtualização começam pelo aproveitamento do *hardware* ocioso nos equipamentos, não havendo necessidade de aquisição de novos dispositivos, assim reduzindo custos. Outro benefício está na otimização do tempo de produção e em criação de ambiente de testes. A maior parte dos dispositivos de *backup* e segurança suportam aplicações em servidores físicos instalados como VMs.

A virtualização oferece um ambiente de testes viável de forma segura e

confiável, facilitando a transferência para o ambiente produtivo, permitindo então a criação de vários ambientes em um único *hardware*. Para que a virtualização ocorra de maneira correta deve ser seguido um conjunto de características. Começando pela característica de particionamento, onde a capacidade partilha o *hardware* físico.

Característica de isolamento, onde ocorre a separação da execução das máquinas virtuais da máquina física. A característica de encapsulamento, onde a virtualização é tratada como um conjunto de arquivos, podendo ser armazenado em vários dispositivos de armazenamento.

Outro benefício trazido pela virtualização é a forma como o sistema pode se recuperar de falhas, assim não interrompendo as atividades do negócio protegendo processos críticos desastres. Esta recuperação se dá através de cópia das máquinas virtuais já criadas, recuperando-as através de pontos de restauração.

2.3.0.1 Linux Contêiner

Segundo Felter et al. (2015), para oferecer melhor desempenho e mais segurança, os múltiplos sistemas *Linux* disponibilizam agilidade e flexibilidade na criação e implementação de aplicações, é um método de virtualização em nível de sistema operacional que permite executar múltiplos sistemas denominados *contêineres* usando um único *kernel*. A tecnologia permite dividir o sistema operacional em diversas máquinas virtuais, ajudando a enfrentar os problemas e os desafios com a virtualização. Pode-se criar ambientes de testes e utilizar o LXC de forma ágil e segura.

O *Linux Contêiner* compartilha o mesmo núcleo que o *kernel* da máquina, para criar diferentes contêineres do sistema operacional, usando modelos que são *scripts* para inicializar o sistema específico.

Os processos que estão sendo executados dentro dos *contêineres* demonstram características que remetem estar sendo executados em um sistema

Linux normal, embora estejam compartilhando o *kernel* subjacente com processos localizados em outros *namespaces*.

O subsistema de grupos de controle do *Linux* ou *cgroups* é usado para agrupar processos e gerenciar seu consumo agregado de recursos. É usado para limitar a memória ao consumo de CPU, fornecendo uma maneira confiável de entregar todos os processos dentro de um contêiner. Como um sistema *Linux* tem apenas um *kernel*, e este *kernel* tem visibilidade total nos *contêineres*, existe apenas um nível de alocação de recursos e agendamento.

Segundo Felter et al. (2015), o LXC trata-se de uma tecnologia de contêineres para Linux, utilizando-se de cgroups para controlar os recursos dos sistema e de namespaces para criar e isolar os objetos dentro do contêiner. A principal característica do contêiner LXC está em ser uma interface de espaço do usuário para o Kernel do Linux, utilizando-se de APIs e ferramentas, as quais fornecem aos usuários Linux a criação de sistemas de gerenciamento. A utilização de virtualizadores requer a emulação do hardware físico, diminuindo o desempenho, já o LXC fornece os recursos ou o mesmo ambiente, evitando a sobrecarga causada pelo virtualizador.

O LXC utiliza o mesmo kernel Linux, não sendo necessário o uso de um segundo kernel, através de uma combinação de recursos de segurança do kernel, se faz possível criar um ambiente virtual na mesma máquina sem um *hipervisor*.

2.3.0.2 Daemon LXD

A arquitetura oferecida pelo LXD contém uma *API REST* para conduzir os contêineres de sistemas completos exatamente como serão dirigidas as máquinas virtuais. Pelo fato do LXD ser baseado no LXC, ele usa a API LXC estável para fazer todo o gerenciamento de *contêineres* atrás da cena, adicionando a *API REST* no topo e fornecendo uma experiência de usuário muito mais simples e consistente. O LXD executa uma cópia sem nenhum programa ou aplicação, de uma distribuição

Linux ou de um computador completo.

Segundo Kedrowitsch et al. (2017), O LXD é o daemon de gerenciamento do LXC desenvolvido pela Canonical, que facilita o gerenciamento de contêineres LXC. Um *contêiner Linux* é um agrupamento de processos que está isolado do resto do sistema através do uso de recursos de segurança do *kernel Linux*, como *namespaces* e grupos de controle. É uma construção semelhante à uma máquina virtual, mas é muito mais leve, sendo que não tem a sobrecarga de executar um *kernel* adicional ou simular o *hardware*. Isso significa que pode facilmente criar múltiplos *contêineres* no mesmo servidor.

2.3.0.3 KVM - Kernel-based Virtual Machine

O KVM suporta dispositivos de entradas e saída emulados através de *QEMU*. A combinação de aceleração de *hardware* e entrada e saída para virtual foi projetada para reduzir a sobrecarga de virtualização para níveis baixos. O KVM suporta migração ao vivo, permitindo que servidores físicos ou mesmo centros de dados inteiros sejam retirados para manutenção sem interromper o sistema operacional convidado. O KVM também pode ser utilizado através de ferramentas de gerenciamento.

De acordo com Chirammal, Mukhedkar e Vettathu (2016), o *hipervisor* baseado no Kernel (KVM) tem como característica também o código aberto, trabalhando com o *hardware* disponível atualmente. Quando um módulo KVM é instalado, o kernel do Linux e assumido como um *hipervisor*.

A capacidade de utilizar pouco espaço em disco se faz através da grande reutilização dos recursos do *hardware*. O KVM necessita de componentes que trabalham juntos para fornecer ao sistema operacional virtualizado.

Segundo Berrangé (2018) o Virt-Manager trata-se de uma interface de usuário no *desktop* para gerenciamento das máquinas virtuais através do componente libvirt. Esta ferramenta é combinada com o KVM, tendo como função for-

necer uma breve visão geral das aplicações em execução, desempenho em tempo real e utilização de recursos estáticos.

Conforme Hat (2018), o Libvirt trata-se de uma coleção de *software* responsável pelo fornecimento do gerenciamento das máquinas virtuais de uma forma sútil aos recursos de virtualização, tais como armazenamento e gerenciamento de comandos.

De acordo com QEMU (2017), o QEMU é responsável por auxiliar o kernel do KVM a criar máquinas virtuais, o qual pode ser utilizado como um emulador de máquina, executando os diferentes sistemas operacionais em diferentes máquinas. Quando utilizado como virtualizador, o QEMU alcança o desempenho próximo ao ambiente nativo, através da execução do kernel convidado diretamente na CPU do *host* hospedeiro.

Segundo Goto (2011), o KVM *Kernel Module*, como o próprio nome já diz, trata-se de um módulo do kernel Linux que funciona com saídas de virtualização dos sistemas operacionais convidados executando suas instruções de entrada da máquina virtual.

O EPT (*Extended Page Table*) é responsável por traduzir os endereços fornecidos pela CPU, permitindo que esta função seja executada de forma automática, sendo assim possível obter desempenho significativamente melhor com a máquina virtual.

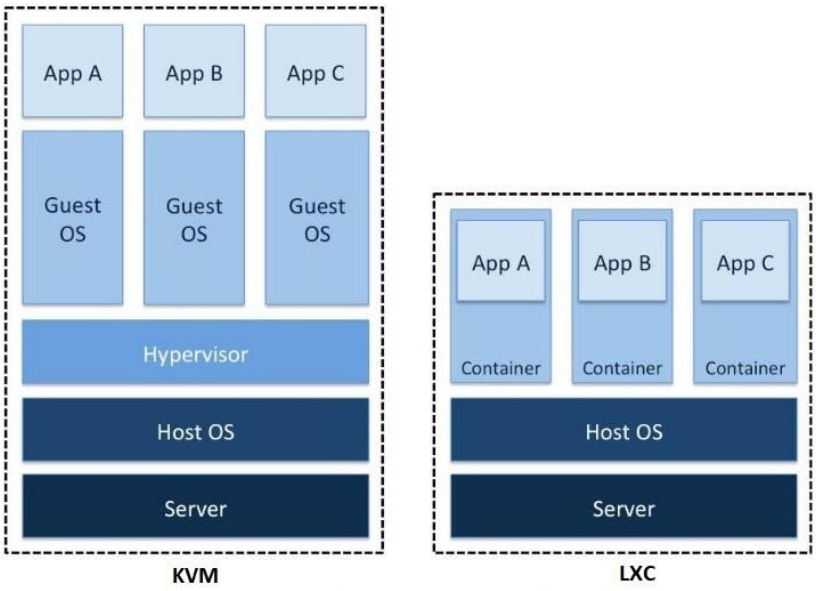
O Virtio é um mecanismo adotado para reduzir a sobrecarga associada ao QEMU e melhorar o processamento com grande desempenho.

O KSM (*Kernel Same-page Merging*) tem por função monitorar o uso da memória dos processos e mesclar as páginas duplicadas em uma única página comum, assim refletindo em economia de memória.

Como demonstrado na Figura 2.10, o KVM atua sobre o sistema operacio-

nal criando um outro kernel, isolando-o completamente, onde cria-se as máquinas virtuais. O LXC é um tipo de virtualização por conteiner no qual apenas cria-se os conteiers utilizando o mesmo kernel do linux.

Figura 2.10: KVM e LXC



Fonte: Adaptado de Beserra et al. (2015)

2.4 *BARE-METAL*

Segundo Giorgi (2013), *bare-metal* caracteriza-se por ambientes de TI onde o sistema operacional é instalado diretamente no *host*, atuando diretamente no *hardware* físico, controlando e monitorando os recursos disponíveis. Por não haver um sistema hospedeiro, há diminuição de concorrências, ou seja, não havendo disputa por recursos físicos e priorização de processos, aumentando a performance dos serviços utilizados. Conforme a Figura SALVO (2014) demonstra o esquema de sequência lógica de *bare-metal*.

Este modelo oferece mais controle ao usuário do servidor tanto como diminuição da latência no acesso aos dados, pelo fato de evitar as etapas de processamento detidas pelo software, como por exemplo, virtualizadores. Sua utilização se faz necessária a aplicações que necessitam maior desempenho e velocidade, trazidos por benefícios como maior processamento de entrada e saída (*input/output*)

Figura 2.11: Bare-Metal

Fonte: Adaptado de SALVO (2014)

por segundo, configuração de *hardware* customizável, ganho maior em consistência de discos e rede principalmente para entradas e saída (I/O).

2.5 MÉTRICAS DE REDE

A utilização de métricas de rede tem por finalidade coletar dados, os quais avaliam o desempenho da rede demonstrando parâmetros qualificando positivamente ou negativamente a rede. Este resultados são utilizados identificar problemas bem como seu tratamento. Nas seções abaixo descreve-se algumas métricas de rede e suas funções.

2.5.1 Latência

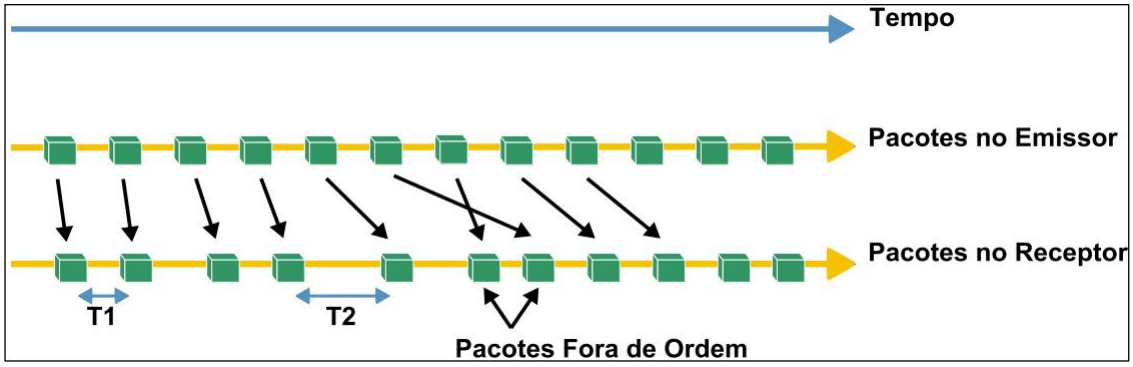
De acordo com Costa (2008), como principal característica, a latência define o tempo total que é gasto por um pacote no caminho entre a origem e o destino. Para calcular este parâmetro, são somados todos os atrasos de processamento de todos os elementos que consistem a rede. Para coletar os parâmetros, um pacote de teste é enviado contendo uma marca de tempo *(timestamp)*. Quando o pacote

é recebido no destino é enviado de volta ao receptor para que ocorra a análise do atraso de ida e volta.

2.5.2 Variação de Latência *(Jitter)*

O *Jitter* tem por característica a variação do tempo de chegada dos pacotes ao endereço de origem, onde a taxa de transmissão varia de acordo com as rotas e meios por onde o pacote é transmitido, diminuindo a qualidade do serviço. A Figura 2.12 demonstra as variações que ocorrem na transmissão dos pacotes, como variações de tempo e desordem dos pacotes. O calculo do *Jitter* segue a seguinte lógica, o tempo de envio é decrementado ao tempo de resposta. Por exemplo: *Jitter = (T2 - R2) - (T1 - R1)*, onde T neste exemplo seria o tempo de envio e R o tempo de resposta.

Figura 2.12: Jitter



Fonte: Adaptado de Costa (2008)

2.5.3 Taxa de Transferência (Vazão)

Conforme Costa (2008), erroneamente confunde-se o desempenho da rede através da sua velocidade. A vazão dos dados demonstra o máximo de dados transportado pela rede do destino de origem ao destino de chegada. Outro fator a ser considerado na medição de vazão é a qualidade, onde os gestores de redes definem uma taxa aceitável de perda, variando então, as particularidades de cada medição. Para as redes *Ethernet* a banda máxima está definida pelo tráfego de

dados de cada equipamento, sendo de 10, 100 ou 1000 Mbit/s. Para calcular a taxa de bits pode haver a conversão para Mbits/s:

$$Taxa_bits = (frame - len/(preramb + frame_len + igp)) * 100 \quad (2.1)$$

Para obter-se o taxa deve ser seguido o seguinte cálculo:

$$igp = ((frame_len * 100)/taxa_bits) - preamb - framem_len \quad (2.2)$$

No Quadro 2.2 consta a vazão máxima para cada tamanho de pacote.

Quadro 2.2: Vazão máxima

Sistema de 10 Mbit/s

Tamanho do Quadro	Vazão de Dados	Preâmbulo e IGP	Quadros por Segundo
64 bytes	7,62 Mbit/s	2,38 Mbit/s	14,88
128 bytes	8,65 Mbit/s	1,35 Mbit/s	8,45
256 bytes	9,28 Mbit/s	0,72 Mbit/s	4,53
512 bytes	9,62 Mbit/s	0,38 Mbit/s	2,35
1024 bytes	9,81 Mbit/s	0,19 Mbit/s	1,20
1280 bytes	9,85 Mbit/s	0,15 Mbit/s	0,96
1518 bytes	9,87 Mbit/s	0,13 Mbit/s	0,81
1522 bytes	9,87 Mbit/s	0,13 Mbit/s	0,81

Sistema de 100 Mbit/s

Tamanho do Quadro	Vazão de Dados	Preâmbulo e IGP	Quadros por Segundo
64 bytes	76,19 Mbit/s	23,81 Mbit/s	148,81
128 bytes	86,49 Mbit/s	13,51 Mbit/s	84,46
256 bytes	92,75 Mbit/s	7,25 Mbit/s	45,29
512 bytes	96,24 Mbit/s	3,76 Mbit/s	23,50
1024 bytes	98,08 Mbit/s	1,92 Mbit/s	11,97
1280 bytes	98,46 Mbit/s	1,54 Mbit/s	9,62
1518 bytes	98,70 Mbit/s	1,30 Mbit/s	8,13
1522 bytes	98,70 Mbit/s	1,30 Mbit/s	8,11

Sistema de 1000 Mbit/s

Tamanho do Quadro	Vazão de Dados	Preâmbulo e IGP	Quadros por Segundo
64 bytes	761,90 Mbit/s	238,10 Mbit/s	1488,10
128 bytes	864,86 Mbit/s	135,14 Mbit/s	844,59
256 bytes	927,54 Mbit/s	72,46 Mbit/s	452,90
512 bytes	962,41 Mbit/s	37,59 Mbit/s	234,96
1024 bytes	980,84 Mbit/s	19,16 Mbit/s	119,73
1280 bytes	984,62 Mbit/s	15,38 Mbit/s	96,15
1518 bytes	987,00 Mbit/s	13,00 Mbit/s	81,27
1522 bytes	987,03 Mbit/s	12,97 Mbit/s	81,06

Fonte: Costa (2008).

Assim, pode-se ver a taxa de transferência dos dados, demonstrando as taxas de aceitação entre a origem o destino. E na mesma forma segue uma taxa de aceitação caso aconetecer alguma perda dos dados.

2.5.4 *Bandwidth*

O *bandwidth*, tem por característica a medida de de capacidade de transmissão da conexão da rede. Desta forma, é determinada a velocidade em que os dados trafegam em uma rede específica.

2.5.5 *Throughput*

De acordo com Costa (2008), esta métrica de rede é diretamente associada a taxa de transferência, calculando a taxa teórica máxima. As variações na medição são alteradas conforme a interferência de alguns fatores, tais como a latência.

2.6 *BENCHMARK*

A utilização de *benchmarks* normalmente se faz para comparar dois ou mais sistemas aplicando medições. Estas medições devem ser realizadas por métodos definidos pela *IETF* (*Internet Engineering Task Force*), órgão o qual é responsável pela padronização dos protocolos.

2.6.1 NetPerf

Segundo NetPerf (2017), com ênfase no desempenho de comunicação na rede, o *benchmark* NetPerf tem como característica avaliar o tempo entre requisições de respostas entre *client* e *server* utilizando-se de *sockets* TCP e UDP, obtendo então a vazão total (*throughput*).

2.6.2 Iperf

Conforme Iperf (2017), o *benchmark Iperf* também utiliza-se dos protocolos TCP e UDP com a finalidade de monitorar estes dois protocolos medindo a largura de banda da rede. Também é possível obter o tamanho do UDP dos dados de leitura. Já utilizando o *Iperf* configurado como UDP é possível calcular a perda de pacote durante o período de avaliação, calcular o *Jitter*.

2.6.3 Uperf

De acordo com Uperf (2018), este *benchmark* trata-se de uma ferramenta de avaliação desempenho de rede, que suporta a modelagem e repetição de vários padrões de rede. Inicialmente desenvolvido pela *Sun Microsystems*, agora está aberto ao público, compatível com os sistemas operacionais *Linux*, *Android*, *Solaris*, *FreeBSD* e*MacOS* . Seu modelo, ao contrário dos demais *benchmarks* que buscam executar referências fixas ou cargas de trabalho, é voltada a fornecer uma descrição da carga de trabalho e esta ferramenta gera a carga de acordo com o modelo.

Após carregar a carga de trabalho deste modelo, pode-se alterar a escala da carga de trabalho, alterar parâmetros, alterar protocolos, analisar os efeitos destas mudanças no modelo. O *Uperf* analisa a largura e avalia o desempenho de banda e latência, tanto direcional quanto bidirecional, utilizando-se de vários protocolos como TCP, UDP,SCTP, SSL.

2.6.4 NetPIPE

Conforme Pitanga (2004), a principal característica, o *NetPIPE* (*Network Protocol Independent Performance Evaluator*) foi desenvolvido como avaliador de desempenho de redes locais, sendo totalmente independente a protocolos, sendo assim possível monitorar *overheads* das camadas dos protocolos superiores.

Com o *NetPippe* são realizados testes do tipo *ping-pong* entre *cliente-servidor*, aonde são realizadas alterações nos tamanhos das mensagens constantemente em tamanhos aleatórios, provendo interferências.

2.6.5 Hpcbench

De acordo com Huang, Bauer e Katchabaw (2005) o desenvolvimento do *benchmark Hpcbench* teve como objetivo medir as redes de comunicação de alta velocidade e baixa latência em sistemas *HPC* baseados em *Linux*.

Como característica, o *Hpcbench* monitora com alta precisão e eficiência, provê suporte para comunicações *UDP*, *TCP* e *MPI*. Os parâmetros de comunicação são ajustáveis de acordo com a necessidade do usuário e particularidades dos ambientes *HPC*. Outra característica é o detalhamento dos resultados de teste.

O *benchmark Hpcbench* foi escrito em linguagem C utilizando *sockets BSD* e *APIs MPI*. Sua composição abrange três conjuntos independentes de *benchmarks* para contemplar a medição dos protocolos *UDP*, *TCP* e *MPI*. Por sua implementação ser baseada em uma infraestrutura comum, a comparação dos resultados se torna fácil com outros protocolos de comunicação.

2.6.6 NOX

O *NOX* é uma proposta de sistema operacional para redes, possuindo como objetivo facilitar o gerenciamento de redes de grande escala. Responsável para fazer os pre-cálculos antes dos dados serem enviados para o destino, intuito de todos os dados seguirem caminhos diferentes para assim chegar mais rápidas no destino. Foi desenvolvido para trabalhar como controlador em redes *OpenFlow*, que em conjunto auxiliam no gerenciamento de uma rede com comunicação de dados.

2.6.7 Drop-In

Uma dinâmica de avaliação dos destinos, antes dos dados serem enviados ao destino, a aplicação *drop-in* faz testes no destino para analisar se é ele que vai receber os dados do remetente ou se o remetente vai ter uma falha na rede, no qual pode impactar possível deficiência para a comunicação.

2.6.8 Multihoming

Oferece a técnica de utilizar múltiplos pontos de conexão a fim de evitar uma parada da rede, se uma das conexões falhar, pode se configurar um computador com mais de uma interface de rede e múltiplos endereços *IP*, conectar um *host* ou uma rede de computadores a mais de uma rede. Isso pode ser feito para aumentar a confiabilidade ou o desempenho, ou para reduzir o custo. Uma maneira de fornecer conectividade de *Internet* aprimorada e confiável sem comprometer o desempenho eficiente.

2.6.9 EWTCP

Um algoritmo desenvolvido em linhas de *Multipath* TCP para troca de dados entre caminhos mais eficientes. *EWTCP* TCP foi desenvolvido para os dados serem enviados em todas linhas disponíveis na rede, mesmo que uma das linhas esta congestionada os dados são enviados mesmo assim. Fraqueza deste algoritmo é óbvia quando os caminhos disponíveis têm *throughputs* diferentes, trocando de linhas como o caso em *WiFi* para GSM ou vice versa.

2.6.10 Shim6

Com o uso de múltiplos prefixos de endereços IP, evita paradas na rede através dos prefixos IPV6, um protocolo que trabalha na camada 3 (camada de rede). Os servidores ou *data centers* que usam em si IPV6, vem consigo alocado Shim6 para fazer a comunicação entre os dados de IPV6 e IPV4. Com a criação de

novas tecnologias, analisa-se a utilização de somente IPV6, e por isso a ferramenta Shim6 é desenvolvida para integrar os protocolos IPs, para que não haja conflitos entre protocolos diferentes.

2.6.11 Host Identity Protocol

Localiza os endereços IP que podem ser encontrados na rede que o *host* esteja conectado. Ele introduz um espaço que identifica os nome de identidade de *host*, com base em uma infraestrutura de segurança de chave pública. O HIP introduz segurança para IPs *Multihoming* e redes de dispositivos moveis. Visando a segurança , os dispositivos ficam mudando seus nomes na rede rapidamente, para assim dificultar mais uma possível invasão maliciosa.

2.6.12 Comparativo entre *Benchmarks*

Como o propósito de decidir qual *benchmark* utilizar, foi desenvolvido uma tabela, a qual demonstra quais pontos positivos de cada *benchmark*, assim tornando mais transparente o que levou-se em conta para escolha. Abaixo encontram-se as comparações entre os *benchmarks*.

Na leitura dos artigos relacionados e também com o estudo da literatura, alguns *benchmarks* foram citados como referência na leitura do desempenho da rede. Em um primeiro momento coletou-se os dados utilizando-se do *benchmark* Iperf3, o qual retornava métricas como *bandwidth* e *throughput*, porém não havendo nenhuma configuração referente a coleta de latência. Assim como o Iperf3, para o *benchmark* NetPerf também não encontrou-se nenhuma forma de filtrar a latência para comprovação dos resultados

A escolha do *benchmark* NetPIPE se deu por suprir a necessidade das métricas necessárias para a validação das hipóteses, sendo eles o *throughput* e latência. Outro fator avaliado para escolha foi a praticidade no tratamentos dos logs gerados nas coletas, assim facilitando a criação de gráficos.

Quadro 2.3: Comparação entre Benchmarks

	Ambiente para testes									Avaliação	
Benchmark	TCP	UDP	SCTP	SSL	MPI	PVM	DLPI	Unix Sockets	Throughput	Latência	
NetPerf	X	X	X					X	X	X	
Iperf	X	X	X							X	X
Uperf	X	X	X	X	X					X	X
NetPIPE	X	X			X		X			X	X
Hpc Bench		X			X	X				X	X
FatTree	X		X			X				X	X
NOX	X	X				X				X	X
DropIn	X		X							X	X
Multihoming	X	X	X							X	X
EWTCP	X	X								X	X
Shim6			X			X				X	
Host Identity Protocol	X					X				X	X
Intranet		X	X			X				X	

2.7 TRABALHOS RELACIONADOS

Conforme foram utilizados os métodos de agregação de link, serão discutidos trabalhos diretamente relacionados os quais utiliza-se de modelos e metodologias explorando esta área.

Como portfólio inicial, o artigo escrito pelos autores Rista et al. (2017), teve como objetivo melhorar o desempenho da rede em instâncias de nuvem OpenNebula baseadas em contêineres criando então uma alternativa viável para a execução de aplicativos *Hadoop* na rede, utilizando-se também de ambientes virtuais como LXC, demonstrando uma redução no tempo de execução de aplicativos *Hadoop* de 33,73%.

O estudo deste trabalho relaciona-se com o nosso estudo, na busca de atender a necessidade da utilização das experimentações utilizando os protocolos IEEE 802.3ad e MPTCP em computação em nuvem, testando seu desempenho em ambiente real e virtualizadores. O artigo acima contribuiu mencionando como utilizar os protocolos de agregação de *link* em instâncias LXC e KVM.

Seguindo o mesmo modelo de experimentação citado pelo autor acima, Matteussi et al. (2014) avaliaram o desempenho em agregações de *links* utilizando-se do algoritmo *Round Robin*, o qual possibilita a agregação de interfaces de rede transmitindo os pacotes em ordem sequencial entre as interfaces.

Este algoritmo, em sua forma original encontrada nas distribuições *Linux*, quando utilizado em agregações de 8 links, ocasionou na perda de 18% do *throughput*, sendo necessário a realização de ajustes diretamente no *Kernel* para chegar-se então ao aumento de 25% de *throughput* para os mesmos 8 links.

Este artigo trouxe como contribuição a necessidade de modificação de alguns parâmetros do *Kernel*, tais como alteração do *buffer*, agregando conhecimento ao nosso trabalho, o qual utilizará também este modelo de configuração de Bonding. O artigo mencionado acima contribuiu demonstrando a forma de configuração do protocolo *bonding*.

Outro artigo relacionado é o de Vogel et al. (2017), o qual buscou apresentar uma avaliação de desempenho para otimizações em cargas de trabalho com uso intensivo de rede. Os resultados apontaram para degradação de desempenho relacionados a nuvem CloudStack usando o KVM, e relacionado ao LXC obteve-se resultados próximo ao nativo. A contribuição deste artigo está em apresentar uma avaliação de desempenho de rede e comparação em relação ao *throughput* de TCP, latência e número de conexões por segundo. Este trabalho contribuiu para a presente pesquisa ao apontar qual o melhor *benchmark* a ser utilizado, o qual apontou o NetPipe como preferível.

Avaliando uma conexão ponta a ponta, Brassil (2005) analisou os pacotes TCP, os quais foram transmitidos entre vários *links*. O modo de configuração *Round Robin* ficou responsável por direcionar os pacotes a cada interface. Porém não obteve-se o resultado desejado, pois os *throughput* alcançado foi apenas de 141Mbs em duas placas de rede *Megabit*.

No trabalho de Brassil (2005) notou-se que a utilização do *bonding* configurado no modo *Round Robin* não obteve resultados satisfatórios, diferenciando-se do nosso trabalho onde é provado que a correta agregação de *link* trás sim resultados favoráveis aos serviços de tráfego de rede. Este trabalho contribuiu citando a queda no desempenho ao utilizar *links* de transmissão heterogêneos, degradando assim o desempenho, sendo necessária a utilização de *links* homogêneos.

Para melhorar o tráfego entre *backbones* e servidores, Hui et al. (2004) buscou através do método *Ethernet Links Bundling Technology* (EIB) aumentar a largura de banda das interfaces dos servidores. Com esta configuração, o *throughput* passou de 94 Mbps para 174 Mbps com os mesmos 36% de processamento do servidor. O estudo realizado pelos autores citados acima contribui com a experimentação de um outro modelo de agregação de *link*, diferentemente do nosso trabalho, que utiliza-se dos protocolos de agregação *bonding* e também do protocolo MPTCP. O artigo acima descrito trás como contribuição a esta pesquisa a redução dos custos utilizando agregação de *link*.

Cho Kenjiro e Crovella (2011) realizaram um estudo amplo com a finalidade de compreender as características de ligação, porém de redes *wireless*. Além disso, investigaram quais fatores influenciam o comportamento da rede, para que o seu desempenho seja maximizado. Os estudos concluíram que os canais mais amplos podem ser explorados através da ligação de canais, porém as decisões de ligação dependem da estrutura dos arredores do transmissor, sendo assim traçado um desenho para que seja explorada toda eficiência do canal, chegando em até 80% de rendimento.

Os autores analisaram o contexto envolto ao cenário da agregação de *link* para então definir um plano de ação para melhor utilizar os recursos que estavam disponíveis. Esta ideia é seguida em nosso ambiente de estudo, no qual otimiza-se os recursos disponíveis explorando o máximo desempenho dos ativos.

Seguindo a mesma linha de pesquisa do autor acima, Deek et al. (2011), utilizou-se da agregação de canal *wireless* para aumentar o rendimento da rede até 80% em relação ao rendimento inicial. O conhecimento da infraestrutura que envolta o campo de testes se faz necessária para operação correta operação da rede agregada de 40MHz. Este estudo contribuiu a presente pesquisa no conhecimento do ambiente envolto as experimentações, no qual acarretou na troca do cabeamento de interligações das máquinas, os quais não estavam de acordo ao *throughput* das interfaces de rede.

Um desafio para Okamoto et al. (2007) foi a implementação de um protocolo RI2N/UDP que aumentasse a tolerância a falhas, principalmente em *clusters*. Pôde-se então aproveitar o máximo desempenho na agregação de duas placas de rede, onde também houve melhoramento na questão de tolerância a falhas. Os resultados apresentados mostram o aproveito de 98% de uso da placa de rede, chegando a uma transmissão de 246 Mbps, aumentando em apenas 2% a sobrecarga de CPU.

A utilização deste protocolo provou a possibilidade de aproveitar ao máximo a utilização das interfaces de rede, contribuindo diretamente ao presente trabalho, que também busca aproveitar ao máximo a utilização das interfaces de rede, porém não apenas limitando-se a agregação de *link bonding*, mas também com a utilização do protocolo MPTCP.

As pesquisas realizadas por Abd et al. (2004) buscam a agregação de *link*, porém de outra forma, utilizando-se do protocolo LSSCTP, o qual agrega a largura de banda disponível monitorando e conectando os caminhos os quais demonstram-se adequados. Os resultados são proporcionais ao número de adaptadores disponíveis para ligação. Este trabalho contribuiu cientificamente desenvolvendo o protocolo LSSCTP, diferentemente do nosso trabalho, que busca aplicar protocolos já desenvolvidos. Em relação ao presente trabalho, a pesquisa proposta por Abd et al. (2004) ressalta a importância dos *links* reagirem de maneira rápida as falhas ocorridas na rede.

Imaizumi et al. (2009) propôs um mecanismo de economia de energia baseado em agregação de *link* 802.3ad, o qual reduziu o consumo energético em 25,4% utilizando o protocolo IEEE 802.3ad e um algoritmo que estima o número apropriado de *links* ativos para suprir a necessidade da agregação de *link*.

Este artigo contribui cientificamente agregando dois meios para o melhoramento do *throughput*, atrelando o protocolo IEEE 802.3ad ao algoritmo, trazendo contribuições ao nosso trabalho através da junção do protocolo IEEE 802.3ad com o algoritmo que estima a necessidade de *links* ativos.

Os autores Barré et al. (2010) implementaram o protocolo MPTCP na plataforma linux, distribuindo um único fluxo TCP por vários caminhos, sem a utilização de aplicativos. O estudo proposto pelos autores visou a funcionalidade do protocolo em toda rede, LAN e Internet, utilizando um servidor remoto e um *smartphone*, os quais demonstraram que o protocolo funciona corretamente em toda a rede.

O propósito deste trabalho caracterizou-se em apenas testar o funcionamento do protocolo, não contribuindo com resultados numéricos. O estudo realizado por Barré et al. (2010) contribui ao presente trabalho do ponto de vista de aplicabilidade do protocolo MPTCP, sendo ele em ambiente de de rede LAN.

Seguindo a linha do autor citado acima, Lim et al. (2014) utilizou outra variante do protocolo MPTCP, o eMPTCP, o qual traz melhorias em relação ao consumo energético consumindo até 15% menos energia por *byte* que o MPTCP. Os autores conseguiram implementar o protocolo sem modificar nenhum aplicativo, tornando viável sua utilização em quaisquer redes e equipamentos.

Utilizando o primeiro contexto no qual o protocolo MPTCP aplica-se ao aumento da largura de banda enquanto o eMPTCP aplica-se ao consumo de energia, podendo ser escolhido o protocolo conforme a necessidade do usuário, contribuindo ao nosso contexto apenas no aumento de *throughput* da rede citando o controle das vazões de rede, dimensionando a transmissão de acordo com a disponibilidade dos fluxos. Este trabalho contribuiu, indicando que outros protocolos podem ser adicionados em juntamente com *Beackmark*, podendo melhorar cada vez mais a eficiência e economia de energia.

Continuando com os trabalhos relacionados, nesta etapa cita-se os trabalhos referentes a TCP *multipath* e outros fatores que fazem avaliação da rede. O artigo Wischik et al. (2011), tem o objetivo é permite que um único fluxo de dados seja dividido em vários caminhos. Isso tem benefícios óbvios para a confiabilidade, e também pode levar a um uso mais eficiente dos recursos em rede. O MPTCP substituído por *DROP-IN Multihoming* escolha de caminhos *COUPLED* e quando o núcleo estiver carregado utiliza *BCube*.

Os testes foram realizados nas aplicações *drop-in*, multihoming e COUPLED, BCube, EWTCP. Analisando qual se beneficiaria em relação ao TCP/MPTCP, para melhor rendimento da rede e confiabilidade na comunicação entre dois hospedeiros. As aplicações que foram testadas, ambas são em relação ao MPTCP, uma é melhor que a outra. Nos testes o EWTCP ficou em primeiro lugar em seguida o MPTCP. Este estudo contribui com nosso trabalho, mostrando a forma que o MPTCP trabalha, permitindo que os dados podem ser enviando em vários caminhos com mais rapidez.

Zhou et al. (2017), mostra que transporte Multipath, ele deve ser implantável na internet atual, sem alterar roteadores, Middlebox ou mesmo redes definidas em NATs. Testes foram realizados com a aplicação Equal Cost Multipath que faz o calculo para ver qual o caminho que é mais eficiente mas tem mas ferramentas que ajudam para realizar os testes.

Inicialmente tem o *handshake*, a pilha verifica se tem vários endereços que possuem rotas para o destino e se não tem tentará abrir sub-fluxos usando endereços atualmente não utilizados, e tem aquele de enviar uma mensagem se o destino esta ligado e uma mensagem de retorno dando a resposta. São usados aplicações, e realizados testes para ver o comportamentos de Multipath de toda internet. Os testes são realizados em *data centers*, comunicações móveis e redes *multi-homed*.

É o Multipath com um aplicação integrada é muito mais eficiente nos trabalhos. Multipath tem avaliações cuidadosas, implementação permite abrir fluxos entre diferentes pares de endereços ou entre os mesmos pares de endereços, nas diferentes portas. Este artigo contribui, que todo o trafego de rede de dados pode ser analisados, verificando se tem gargalos de congestionamentos ou perda de dados. O *handshake* é umas das possibilidades que mostra isso para o usuário.

Barré, Paasch e Bonaventure (2011), tem o objetivo de rede otimizada para alcançar o alto desempenho e relatar as medidas que mostram o desempenho do controle de congestionamento e do congestionamento acoplado, com os testes a

partir do MPTCP no *kernel* do *Linux*. Os dados são produzidos entre um cliente e um servidor, link A e link B, sendo que ombos tem mais rendimento mais que o outro. Os problemas de desempenhos dos dois link geram problemas para MPTCP, quando os *buffer* da rede são em valores diferentes e os atrasos ou perdas dos dados foram maiores que 6%.

Os testes feitos com as aplicações *shim6, Host Identity Protocol, iperf*. Utilizado quatro estações de trabalho Linux. As duas estações de trabalho usam as CPUs Intel R Xeon (2.66GHz, 8GB de RAM) e os Controladores Gigabit Ethernet IntelR 82571EB. O protocolo SCTP permitem que os *hosts* usem vários caminhos ao mesmo tempo. Embora implementado em vários sistemas operacionais, o SCTP ainda não é usado além de aplicativos específicos. Mas os *firewall* ou NAT bloqueia os caminhos que são criados pelo SCTP, esses caminhos que são desconhecidos para eles. Este projeto contribui com nosso trabalho, indicando que a forma de como o *Benchmark* é implementado pode implicar na comunicação, podendo assim perder dados e implicações na comunicação.

Este artigo, a Kheirkhah, Wakeman e Parisis (2015), tem como objetivo melhorar o rendimento, e ficar longe dos caminhos mais congestionados, entregando os dados com qualidade. Foram realizados testes entre dois *links*, primeiro com Eifel Algioithm e outro com SACK Algotithm. Enviou-se 3 dados para o destino e o outro dado ficando na espera, quando há falha no primeiro dado é entregue o segundo. Mas se falhar os 3 dados é enviado aquele outro que não foi enviado. Contribui com o nosso trabalho, na forma de ficar longe dos caminhos congestionados, mas o artigo Kheirkhah, Wakeman e Parisis (2015) mostrou que o uso de algoritmos pode também melhorara comunicação ou troca de dados.

Oferecer um rendimento melhor e uma melhor equidade. O objetivo é mitigar o colapso do incast, permitindo que vários subfluxos MPTCP competem de forma justa com um fluxo de TCP. Com *Ming et al. (2014)*, foi realizados testes sobre o MPTCP para competir de forma justa com um único fluxo de TCP no gargalo compartilhado. Seguindo a forma de TCP, os testes do MPTCP tem a fim de

aproveitar de forma eficaz e perfeita a vantagem de vários caminhos para fornecer melhor rendimento e melhor equidade. O autor mostram que o aumento de números de servidores conectados pode ter um padrão diferente na comunicação. Por isso foi projetado EW-MPTCP para a avaliação do MPTCP. Temos a contribuição ao nosso trabalho, que os subfluxos do MPTCP competem para entregar os dados, o fluxo que estiver congestionado vai dando caminho para os outros fluxos disponíveis, mas assim que liberar os fluxos bloqueados, eles são enviados e entregues ao destino.

Para um controle de tabelas baseadas em Openflow, Mattos et al. (2011) calcular vários caminhos e evitar formação de laços, calculando múltiplos caminhos entre origem e o destino. Aumentando assim a taxa de transferência e também a eficiência da banda passante disponível. Para pre-calcular um determinado numero de caminhos diferentes e de menor custo entre pares de nos, foi usados os *bechmarks* NOX, BCube, FatTree. Para a criação de multiplos caminhos é proposto a criação de VLAN para identificar e distribuir o tráfego. Um ambiente para emulação é criado mininet para executar os protocolos Openflow. A maquina virtual foi configurada com 64 GB de memoria RAM e acessando 16 nucleos de processamento. A maquina fısica que hospeda a maquina virtual e um Intel Xeon X5570 de 2.93 GHz , com 96 GB de memoria. Todos os comutadores, servidores e o controlador de rede NOX executam na maquina virtual. O Mininet instancia um controlador real, executando a aplicação desenvolvida, comutadores em software e cria ambientes virtuais que exercem a fincão de servidores.

São rede otimizada para alcançar o alto desempenho e relatar as medidas que mostram o desempenho do controle de congestionamento e do congestionamento acoplado.Barré, Paasch e Bonaventure (2011) é uma implementação do MPTCP no kernel do Linux, usando os *benchmarks* shim6, Host Identity Protocol, iperf. Os dados são produzidos entre um cliente e um servidor, link A e link B, sendo que ombos tem mais rendimento mais que o outro. Os problemas de desempenhos dos dois link geram problemas para MPTCP, quando os buffer da rede são em valores diferentes e os atrasos ou perdas dos dados foram maiores que

6%. Utilizado quatro estações de trabalho Linux. As duas estações de trabalho à esquerda usam as CPUs Intel R Xeon (2.66GHz, 8GB de RAM) e os Controladores Gigabit Ethernet Intel R 82571EB.

A diferença das duas aplicações é que um pode ser agregada com o banco de dados com o *Google* para sempre ter retransmissão dos dados quando acontecer a falha. Quando o atraso de ponta a ponta, variar muito, os pacotes podem chegar fora de sequência. Ser o caso em redes sem fio onde os dispositivos móveis podem mudar o ponto de acesso usado para acessar a Internet. A reordenação de um pacote faz com que o receptor responda com confirmações duplicadas, e isso pode induzir o remetente a inferir erroneamente uma perda de pacotes. A ideia do autor é usar simultaneamente vários caminhos, para melhorar a robustez e o desempenho das conexões de ponta a ponta. Contribuiu com o nosso trabalho com a relação do MPTCP tem a melhor eficiência nos envio ou comunicação os dados. Difícil acontecer perda de dados, pois os dados que são bloqueados devido o congestionamento da rede, sempre no fim eles são enviados. O envio não acontece caso a internet ficar indisponível.

Andrioli, Rosa Righi e Aubin (2017) tem o objetivo de oferecer uma qualidade de serviços sobre SDN, balancear a carga dos dados e controlar a ocorrência de congestionamentos. A SDN é um controle da rede, onde um controlador programável é responsável por gerenciar regras para o encaminhamento dos dados para diversos dispositivos. Um controlador programável e flexível, O SDN pode solucionar problemas na otimização do tráfego de dados.

Contribui com o nosso trabalho, de fato por gerenciar regras para o encaminhamento dos dados para diversos dispositivos, com o comportamento da rede que seja mais flexível, bem como adaptável e gerenciável. Assim pode acontecer com o MPTCP, os dados são enviados de acordo com a disponibilidade dos *link* que enviam os dados.

Lima (2017), um protocolo mais eficiente para o aumento da vazão fim-a-fim através da agregação da transmissão de dados TCP, maior tolerância a falhas

de comunicação, uma vez que, na indisponibilidade de um sub-fluxo, há um ou mais sub-fluxos ativos. Escalonamento de pacotes, controle de congestionamento e gerenciamento de caminhos. A implantação para os testes foi Single-mesh, é mitigar o congestionamento inter-caminho, aumentando a taxa de transmissão dos sub-fluxos em caminhos não congestionados. Tem a contribuição ao nosso trabalho, com os escalonamento de dados utilizando o MPTCP, com tolerância a falhas e aumento de transmissão na troca de dados.

Multipath com um outro conjunto realizando os trabalhos, pode ser mais eficientes que o MPTCP por si só. Os ganhos pode ser mesmo de 1%, mas mostra eficiência em relação a qualquer outro desempenho da rede. Cria um ambientes controlado com flexibilidade sobre domínios definidos por softwares e avaliar e realizar os desempenhos máximos que o MPTCP pode utilizar em varias condições para a internet. Autor da exemplos de gerenciadores de dados e diz que o *Single-smash* é mais eficiente para controladores de congestionamento. Contribui com o nosso trabalha, sobre a utilização dos desempenho definidos sobre o MPTCP. O controle de envio dos dados na rede pode ser definido de acordo com a necessidade do usuário,

O artigo Van Der Pol et al. (2012) demonstra que, a troca de informações em cima dos protocolos com Multipath, suportar fluxos de dados de computação de alto desempenho (HPC) usando os recursos do OpenFlow. é uma maneira eficiente, mas caso de falha quando o tráfego tem muitos fluxos diferentes, o tráfego será distribuído uniformemente pelos vários caminhos, entregando os dados ao remetente, e caso de falha, a os dados são reenviados. Temo relação com o nosso trabalho, sobre a eficiência de atribui os dados pelo sub-fluxos disponível na rede, evitando a lentidão.

Al-Fares et al. (2010) criado para redes de *data center* ECMP, é um sistema de agendamento de fluxo dinâmico, coleta informações de fluxo de *switches*, calcula caminhos não conflitantes. Maximizar a utilização agregada da rede, largura de banda de bi-secações, e fazê-lo com uma sobrecarga de planejamento mínima

ou impacto nos fluxos ativos. Aplicações que fazem os testes é intranet, PortLand, são interruptor não bloqueador hipotético para vários padrões de comunicação interessantes e realistas, entre em nosso teste até 4 mais largura de banda do que o estado das técnicas ECMP.

Um sistema de agendamento de fluxo dinâmico para topologias de comutação em vários estágios encontrados em *data centers*. Hedera, nome dado a uma agregação, coleta informações de fluxo de *switches* constituintes, calcula caminhos não conflitantes para fluxos e instrui os *switches* para reencaminhar o tráfego de acordo. Hédera oferece uma largura de banda de bi-secção que é 96% do ótimo e até 113% melhor do que os métodos estáticos de balanceamento de carga. Com relação ao MPTCP, os testes usando o Hedera, tem resultados parecidos, pois, com a agregação usando o MPTCP usa as placas que estiver disponível no *host*, mostrando os resultados de 110% melhor ao método normal que é o TCP.

O artigo Shamani et al. (2016), demonstra que os dispositivos moveis foram aumentados de acordo com a utilização do MPTCP, para explorar a diversidade de caminhos que estarão liberados. *Multipath*, leva em conta a qualidade dos sinais, mas um dos problemas do *Multipath* é o alto consumo de energia, mas é trabalhado para tomada de decisões sobre incertezas de consumos elevados, assim selecionando a melhor politica para as eficiências. Este artigo tem relação ao nosso trabalho, pelo motivos de ocupar mais *hardwares* para o uso do MPTCP, isso pode ser um gasto na aquisição. Mas o beneficio seria, com essa agregação das placas, o gasto de *hardware* seria pequeno, mas o ganho na eficiência e afirmações seria duplicada em relação a comunicação usando caminho único TCP.

Zhou et al. (2017), o rendimento é aumentado de acordo com a troca de dados, reduzindo os desperdícios dos dados não acarretando em falhas. É usado vários caminhos para a comunicação, com objetivo que o cliente não fique com problemas, aumentando o seu rendimento na transmissão dos dados. Para o uso do MPTCP, antes de tudo é feito testes para ver o tamanho e qual vai ser o desempenho. A transmissão de vários caminhos muda totalmente o comportamento

da rede, fazendo com que trabalhe muitas vezes mais rápida. O nosso trabalho, usando o MPTCP, também muda o comportamento da rede, faz a escolha das rotas mais eficaz para a troca de informações entre dois *host*, e antes do envio dos dados o MPTCP faz um diagnostico dos caminhos disponíveis para inciar a comunicação.

Sobre a busca e conhecimento dos trabalhos relacionados, ela foi muito importe para a dupla. Foi assim que realmente entendemos os protocolos estudados e encontramos formas diferentes de uso de cada protocolo, explanando a forma com que é utilizado e qual realmente é a função. Mas em estudo para entender os protocolos, encontramos artigos relacionados que fazem parte importantes em relação ao *Multipath* TCP e o *Bonding*, mostrando a implementação em outras. Com as descrições dos trabalhos acima, montou-se uma tabela, com a citação do projeto, qual o objetivo do projeto de pesquisa, qual a tecnologia de virtualização, qual o resultado final do projeto e qual entendimento, e por fim, qual foi a contribuição que o artigo de pesquisa vai nos ajudar para a realização deste projeto.

Quadro 2.4: Trabalhos Relacionados

Título	Objetivo	Benchmark	Tecnologia de Virtualização	Resultados	Contribuições	Ano
Kaup et al. (2015)	Redução de cursto usando MPTCP.		N/A	A implementação atual do MPTCP não é o poder idela e sugestões sobre a melhoria da eficiência energética.	Menos consumos de energia e melhorias no processamento do MPTCP.	2015
Barré et al. (2010)	Analisar o comportamento do MPTCP.	O autor não realizou testes de desempenho, apenas de aplicabilidade.	N/A	O MPTCP muito mais eficiente em comparação com o TCP.	Melhor desempenho da rede com a utilização do MPTCP.	2010
Lim et al. (2014)	Analisar o consumo de energia elevado com o uso do MPTCP.	Realizou-se testes de throughput nos protoclos eMPTCP, MPTCP e TCP.	N/A	O MPTCP é o método que mais consome energia.	Fornece os benefícios do MPTCP como a robustez.	2014
Edin (2011)	Comparar as tecnologias deservidores Azure e Amazon.		N/A	A virtualização em nuvem tem um quesito principal de segurança egerenciamento de modelos de nuvem, sendo pública, híbrida, privada.	Ferramentas com essas duas tecnologia Amazon e Azure.	2011
Zhai, Cummings e Dong (2008)	Resolução do Live Migrationatravés de dispositivo Pass-Through.	O autor realizou testes de funcionalidade, não avaliando o desempenho.	Implantou-se e testou-se a funcionalidade das tecnologias, provando o seu correto funcionamento.	Integra o dispositivo virtual hotplug ACPI com diver de ligação do Linuxhabilitando a rede contínua para dispositivos NIC atribuídos diretamente.	Hotplug com um dispositivo pass-through passou a funcionar corretamente no Xen. Com isso e o driver de ligação, os convidados do Linux podem fazer Live Migration com sucesso.	2008
Wang e Ng (2010)	Analisar o desempenho em rede das instâncias da máquina virtual e entender o impacto da virtualização no desempenho da rede experimentado pelos usuários.	Analisouo Bandwidth e a latência do ambiente virtual.	N/A	Variações de atraso anormais com compartilhamento de processadores em nuvem.	Uso de Xen para a criação de varias máquinas virtuais em uma máquina física.	2010

Quadro 2.5: Trabalhos Relacionados

Título	Objetivo	*Benchmark*	Nuvem e Virtualizador	Resultados	Contribuições	Ano
Lim et al. (2014)	Desenvolver um modelo de consumo de energia usando o eMPTCP em celulares e WiFi.	Realizou-se um teste de throughput com a largura de banda de 10Mbps e 1Mbps em um arquivo de 256MB.	N/A	O eMPTCP consome 15% menos energia por byte do que MPTCP. Utilizou 25% menos dados.	Os autores implementaram o protocolo MPTCP de modo funcional.	2014
Rista et al. (2017)	Melhorar o desempenho da rede em instâncias de nuvem com base em *container* para executar aplicativos Hadoop na rede.	TestDFSIO, Netpipe .	OpenNebula com LXC	Hadoop houve redução do tempo de execução em 33,73%.	O padrão IEEE 802.3ad aumenta a largura de banda em instâncias de nuvem LXC em clusters Hadoop.	2017
Abd et al. (2004)	Propor uma extensão para SCTP, o LSSCTP, que agrega a largura de banda.	Para medição foram realizados testes de transferência, com a largura de banda definida em 1,5Mbps com RTT de 1/4 100 ms.	OPNET	O estudo mostra que o LS-SCTP é capaz de lidar de forma eficiente com falhas no curto prazo, bem como a longo prazo.	Ao modificar o SCTP para o LS-SCTP permitiu-se o compartilhamento de carga.	2004
Imaizumi et al. (2009)	Propor redução energética utilizando a agregação de link 802.3ad.	Simulação com dados de rastreamento de tráfego real, utilizou-se tráfego unidirecional.	N/A	Redução do consumo de energia em 17,7 W. O algoritmo proposto reduz até 25,4% (6,7 W fora de 26,4W).	Os autores propuseram um algoritmo que reduz em aproximadamente 25,4% com atraso de *buffer* e variância de atraso de *buffer*.	2009
Lim et al. (2015)	Projetar, implantar e avaliar o eMPTCP, com o objetivo de reduzir o consumo de energia.	Mediu-se desempenho do eMPTCP ao baixar arquivos de diferentes tamanhos.	N/A	O eMPTCP reduz o consumo de energia em até 90% para arquivos pequenos e até 50% para arquivos grandes.	Os autores realizando experimentos com o eMPTCP provando que este protocolo é capaz de reduzir o consumo energético.	2015
Deek et al. (2011)	Compreender as características da ligação de canal nas redes 802.11n e os fatores que influenciam esse comportamento para prever o comportamento.	Iperf	N/A	Foram exploramos todo o desempenho para a ligação de canais e melhorar o rendimento da rede em mais de 80%.	O ambiente físico influencia na incorporação da operação de 40MHz em implantações de rede para maximizar o desempenho e a eficiência.	2011

Quadro 2.6: Trabalhos Relacionados

Título	Objetivo	*Benchmark*	Nuvem e Virtualizador	Resultados	Contribuições	Ano
Okamoto et al. (2007)	Implementar e avaliar o desempenho de um RI2N/UDP avaliando a largura de banda e tolerância a falhas para clusters de PC.	A tolerância a falhas foi testada interrompendo o *link*, o *throughput* foi avaliado com transferência através de testes entre 2 *hosts* 10 vezes.	N/A	A implementação do RI2N/UDP fornece *throughput* agregados. O rendimento RI2N/UDP é de 246 MB/s, onde é alcançado 98%.	Os autores desenvolveram o modelo RI2N/UDP dobrando a largura de banda com aumento de 2% de sobrecarga.	2007
Matteussi et al. (2014)	Avaliar o desempenho de agregações de *link* utilizando o algoritmo *Round-Robin*.	Iperf	CoreEmu, containers.	18% de perda de *throughput* com 8 agregações. Ajustando o *Kernel* aumentou-se 25% o *throughput* para 6,7 e 8 *links*.	Os autores modificaram o *kernel* afim de realizar ajustes para suprir um desvio quando implementado agregações entre dois *links*.	2014
Hui et al. (2004)	Melhoramento do tráfego entre backbones e servidores aumentado a largura de banda com o método *Ethernet Links Bundling Technology* (EIB).	Netperf	N/A	Aumento da largura de banda para 176Mbps, mantendo a mesma carga da CPU de 38%.	Os autores propuseram o método Ethernet Links Bundling Technology (EIB) para obter-se maior desempenho nos níveis de servidor.	2004
Brassil (2005)	Examinar o desempenho de ponta a ponta de uma única conexão TCP com pacotes transmitidos através de vários *links*.	Ferramenta de teste TTCP, Equalizador de Tráfego (TEQL)	N/A	*Throughput* de 141Mbs em duas placas de rede Megabit.	Os autores provaram que o TCP em múltiplos canais de comunicação heterogênea degrada o desempenho da rede.	2005
Luo (2010)	Apresentar tecnologias de virtualização de E/S de rede.	Não foram realizados testes de desempenho, somente de funcionalidade.	N/A	Os resultados experimentais mostram uma redução de 39% na latência em comparação com um switch OpenFlow.	Vantagens de desempenho, isolamento de Maquinas Virtuais.	2010

Quadro 2.7: Trabalhos Relacionados

Título	Objetivo	Benchmark	Nuvem e virtualizador	Resultados	Contribuições	Ano
Kheirkhah, Wakeman e Parisis (2015)	Melhorar o rendimento e qualidade nos sub-fluxos MPTCP.	Realizado diagnostico da rede.	N/A	O MPTCP utiliza dois modos de de sequencias 64bits para o reordenamento e perda e 32bits para alocação de nível de MTPCP para TCP.	A ideia do autor é usar simultaneamente vários caminhos.	2011
Ming et al. (2014)	Oferecer um rendimento para o MPTCP de forma igual com um fluxo de TCP.	Equalizador Fat-Tree, aproveita de forma eficaz a vantagem de vários caminhos.	N/A	*Multipath* TCP é mais eficiente que o TCP.	O aumento de números de servidores conectados pode ter um padrão diferente na comunicação.	2014
Mattos et al. (2011)	Controlar as tabelas de Openflow, para calcular os caminhos e evitar formação de laços.	Testes para calcular um determinado número de caminhos diferentes, usando NOX, BCube, FatTree.	N/A	Usar caminhos redundantes, melhorando a taxa de transferência em até 50%.	Autor demonstra uma rede de Centro de Dados atualmente, tentando conciliar redundância, escalabilidade e eficiência de custo.	2014
Barré, Paasch e Bonaventure (2011)	Rede otimizada para alcançar o alto desempenho e relatar as medidas de congestionamento.	Testes realizados com shim6, HIP, iperf, permitindo o uso dos dados em vários caminhos.	N/A	Criado as rotas com os uso das aplicações, muitas delas não permitidas, pelo bloqueio dos firewall ou Nat de uma rede.	Autor verifica varias extensão de SCTP que foram criaadas para que o TCP use vários caminhos.	2016
Lima (2017)	Um protocolo eficiente para o aumento da vazão fim-a-fim através da agregação da transmissão de dados TCP.	N/A	N/A	Single-smash realizando os trabalhos, pode ser mais eficientes que o MPTCP por si só.	Autor da exemplos de gerenciadores de dados e diz que o Single-smash é mais eficiente para controladores de congestionamento.	2017

Quadro 2.8: Trabalhos Relacionados

Título	Objetivo	Benchmark	Nuvem e Virtualizador	Resultados	Contribuições	Ano
Tudoran et al. (2012)	Uma extensões para TCP que permite o espalhamento de um único fluxo em vários sub-fluxos.	Implantação do multipath para aumentar e melhorar a resiliência e equilibrar o congestionamento na rede.	N/A	MCTCP pode agendar o tráfego, reduzindo o congestionamento em um único caminho.	O MPTCP e sub-fluxos com a troca de informações pode ter gargalos e ao invés de beneficiar a comunicação pode prejudicar.	2011
Zhou et al. (2017)	Mostra que a transmissão de múltiplos caminhos pode efetivamente aumentar o rendimento da carga útil.	Usado NORNET da Internet, testando o tamanho real e analisar o desempenho do MPTCP.	TESTBED NORNET CORE.	Múltiplos caminhos pode efetivamente aumentar o rendimento da carga útil do aplicativo.	Tamanho do buffer podem aumentar o rendimento da carga útil, e poucas chances de perda dos dados.	2017
Vogel et al. (2017)	Apresentar um desempenho para possíveis otimizações de cargas com uso intensivo de rede.	NetPipe, Uperf.	LXC, KVM, CloudStack.	Os resultados apontam para degradação de desempenho a nuvem OpenStack é KVM. Já o LXC obteve resultados próximos ao nativo.	Avaliação de desempenho de rede e comparação em relação ao throughput de TCP, latência e número de conexões por segundo.	2017
Andrioli, Rosa Righi e Aubin (2017)	Oferecer uma qualidade de serviços sobre SDN, balancear a carga dos dados e controlar a ocorrência de congestionamentos.	N/A	N/A	A SDN é um controle da rede, onde um controlador programável é responsável por gerenciar regras para o encaminhamento dos dados.	Um controlador eficiente, flexível. O SDN solucionar problemas na otimização do tráfego de dados.	2017
Shi et al. (2003)	Implantação de recursos e extensões em dispositivos moveis.	Testes realizados para agendamento de pacotes e caminho.	N/A	O Multipath TCP transmite os dados através da interface e aplica o retorno exponencial ao seu temporizador.	O Multipath TCP usa dois níveis de números de seqüência: o número de seqüência regular que rastreia os bytes enviados por este subfluxo.	

Quadro 2.9: Trabalhos Relacionados

Título	Objetivo	Benchmark	Nuvem e Virtualizador	Resultados	Contribuições	Ano
Van Der Pol et al. (2012)	Suportar fluxos de dados de computação de alto desempenho usando OpenFlow	N/A	N/A	Usa as mesmas métricas do TCP, SYN, SYN / ACK, ACK.	O MPTCP detecta e usa vários caminhos para um destino com agendamento de pacotes no uso do OpenFlow em switches.	2012
Van Der Pol et al. (2012)	Busca de caminho mais eficiente para troca de dados, e estabilidade quando ocorrer problema na rede.	Diferentes tecnologias para o acesso a internet, ADSL, Wifi, 3G e LTE	N/A	A comunicação multimídia usa vários caminhos de ponta a ponta para transmitir um dado, evitando o congestionamento e reduzindo a perda.	O protocolo SCTP padrão monitora a disponibilidade de cada caminho e muda a transmissão caso aconteça falha	2002
Shamani et al. (2016)	Utilização de eMPTCP pode reduzir o consumo de energia	N/A	N/A	Uma conexão MPTCP pode alcançar uma maior taxa de transferência do que uma conexão TCP padrão.	O uso de MPTCP é raro ocorrer erros	2014

CAPÍTULO 3: EXPERIMENTOS E RESULTADOS

No decorrer deste capítulo serão discutidos os resultados obtidos nesta pesquisa. Nas próximas sessões serão detalhados os ambiente de testes, as metodologias seguidas nesta pesquisa e os *benchmarks* utilizados na realização dos testes.

3.1 LOCAL DE APLICAÇÃO DO ESTUDO

O LARCC (Laboratório de Pesquisas Avançadas para Computação em Nuvem) é um laboratório voltado para computação em nuvem, focados em modelos de serviço de Infraestrutura como Serviço (IaaS), Plataforma como Serviço (PaaS) e Software como Serviço (SaaS). Outras linhas de pesquisa são oferecidas em sistemas distribuídos, programação para redes, eficiências energética para *data-center*, nuvem privada, mineração de dados e aprendizado de máquinas para agricultura. O laboratório localiza-se na cidade de Três de Maio, na Av. Santa Rosa nas dependências da Sociedade Educacional Três de Maio (SETREM).

O grupo de pesquisa é composto por nove membros. Três com doutorado, dois com mestrado, dois com licenciatura em tecnologia e dois com diploma em progresso. Este grupo é formado por três universidades, SETREM, PUCRS e Unipampa que fornecem recursos humanos para pesquisa. Além disso, o LARCC tem colaboração com as empresas ABASE e Automasul.

O Laboratório é um lugar de pesquisa para a evolução dos acadêmicos, com características de estudar e implementar ferramentas de serviços voltada a

plataforma de nuvem. Essas ferramentas que são testadas e usadas no laboratório podem futuramente ser usadas em empresas, propondo soluções esperadas para possíveis problemas e atendendo a sua perspectiva.

3.2 AMBIENTE DE TESTES

O ambiente utilizado para os testes dividiu-se em duas partes, uma parte está destinada aos testes relacionados ao protocolo *bonding* e a outra destinada aos testes realizados com o protocolo MPTCP.

3.2.1 Ambiente de testes *Bonding*

Para realização dos testes com o protocolo *Bonding* utilizou-se duas máquinas, uma configurada como servidor e outra como cliente. A máquina configurada como servidor dispões de três interfaces de rede, cada interface sendo Gigabit. Uma das interfaces está configurada para acessar a internet, e as restantes para a utilização do protocolo *Bonding*. Já a máquina configurada como cliente dispõe de duas interfaces de rede, ambas destinadas a utilização do protocolo *bonding*.

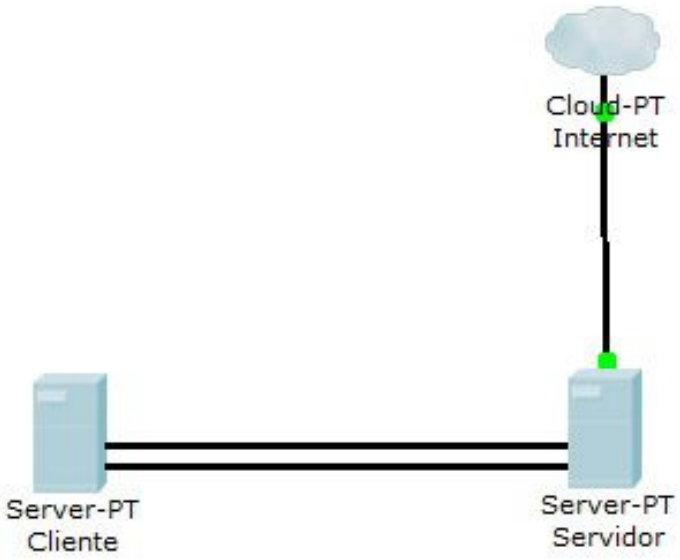
O sistema operacional utilizado para este trabalho é o Ubunto Server 16.04. Abaixo segue o Quadro 3.1 com a especificação do *Hardware* de cada *host*.

Quadro 3.1: Hardware

	Processador	Placas de Rede	Memória RAM
Servidor	Intel(R) Core(TM) i5 CPU, 3.20ghz	3 Interfaces Gigabit	4GB
Cliente	Intel(R) Core(TM) i5 CPU, 3.20ghz	2 Interfaces Gigabit	4GB

As duas máquinas estão interconectadas com cabos de rede CAT6, as placas estão configuradas como escravas para a utilização do protocolo *Bonding* como descrito na Figura 3.1.

Figura 3.1: Ambiente de Testes Bonding



3.2.1.1 Configuração Bonding

Para configuração do protocolo *Bonding*, foram analisados os trabalhos relacionados de Aust et al. (2006) e Matteussi et al. (2014), os quais implantaram e testaram estas configurações. Em um primeiro momento, configurou-se as interfaces de rede de acordo com a configuração sugerida pelos autores citados acima, porém, ocorreram erros de compatibilidade com a versão do sistema operacional utilizado neste trabalho.

Efetuou-se diversos testes até a conclusão da correta configuração das interfaces. Para o servidor, adotou-se o endereço IP 192.168.1.100. Para a interface `ens0p25`, destinou-se a conexão com a Internet. As interfaces `ens128` e `ens132` passaram a ser escravas do protocolo *bonding*. Já para o cliente, configurou-se as duas interfaces como escravas do protocolo *bonding*.

A configuração adotada para as interfaces de rede do servidor encontra-se da seguinte forma no Código 3.1:

Listing 3.1: Configuração Interface Servidor

```
1 source /etc/network/interfaces.d*
2 # The loopeback network interface
```

```
3  auto lo
4  iface lo inet loopback
5  # The primary network interface
6  auto ens0p25
7  iface ens0p25 inet dhcp
8  # The second network interface
9  auto ens128
10 iface ens128 inet manual
11 bond-master bond0
12 # The thrid network interface
13 auto ens132
14 iface ens132 inet manual
15 bond-master bond0
16 auto bond0
17 iface bond0 inet static
18 address 192.168.1.100
19 mask 255.255.255.0
20 gateway 192.168.1.1
21 dns-nameserver 8.8.8.8
22 bridge_ports ens128 ens132
23 bond-slaves  ens128 ens132
24 bond-mode balance-rr
25 bond-mimon 100
26 bond-primary ens128 ens132
```

As linhas 11 e 15 do Código 3.1 foram adicionas para direcionar as interfaces `ens128` e `ens132` como escravas do protocolo *bonding*. Na linha 22 do Código 3.1, adicionou-se as interfaces `ens128` e `ens132` como *bridge* do protocolo *bonding*. A linha 24, `bond-mode` é responsável por definir qual o modo do protocolo *bonding* a ser setado. A linha 25 é responsável por fixar um tempo em milissegundos, no qual, o *kernel* inspeciona o *link* a cada determinado período de tempo. Para configuração referente ao *Bond0* utilizou-se dois IPs, sendo `192.168.1.100` no *host* Servidor e `192.168.1.200` no *host* Cliente.

Para configuração do *host* do cliente, configurou-se as duas interfaces como escravas do *bonding*. A configuração adotada para o cliente para a execução

do `bond0` esta disposta no Código 3.2:

Listing 3.2: Configuração Interface Cliente

```
source /etc/network/interfaces.d*
# The loopeback network interface
auto lo
iface lo inet loopback
# The primary network interface
auto enp0s25
iface enp0s25 inet manual
bond-master bond0
# The second network interface
auto ens132
iface ens132 inet manual
bond-master bond0
auto bond0
iface bond0 inet static
address 192.168.1.200
mask 255.255.255.0
gateway 192.168.1.1
dns-nameserver 8.8.8.8
bridge_ports enp0s25 ens132
bond-slaves enp0s25 ens132
bond-mode balance-rr
bond-mimon 100
bond-primary enp0s25 ens132
```

3.2.1.2 Resultado Bonding aplicado no ambiente nativo

Antes de se iniciar as experimentações sobre agregação de *link*, é importante demonstrar os resultados do ambiente normal de utilização, sem a aplicação do protocolo *bonding*. Na Figura 3.2, notou-se a taxa máxima de *throughput* que é oferecida pela interface, sendo de 887 Mbps e na Figura 3.3 também a latência apresentada de 73 ms.

Os testes no ambiente nativo servem como base de referência para avaliação do desempenho dos ambientes LXC e KVM.

Figura 3.2: *Throughput* NetPipe Ambiente Nativo Sem *Bonding*

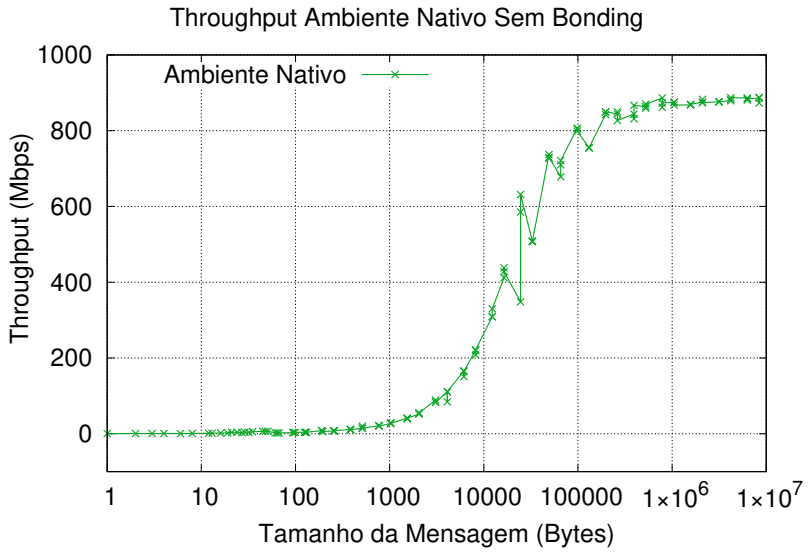
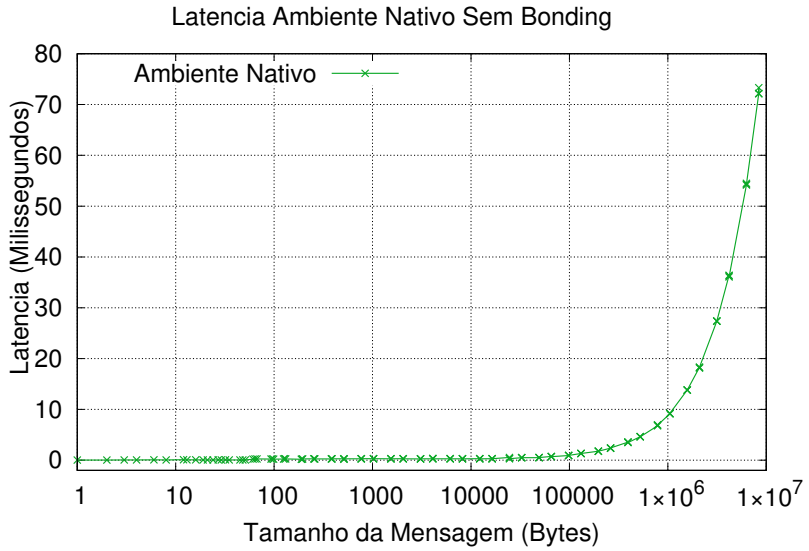


Figura 3.3: Latência NetPipe Ambiente Nativo Sem *Bonding*

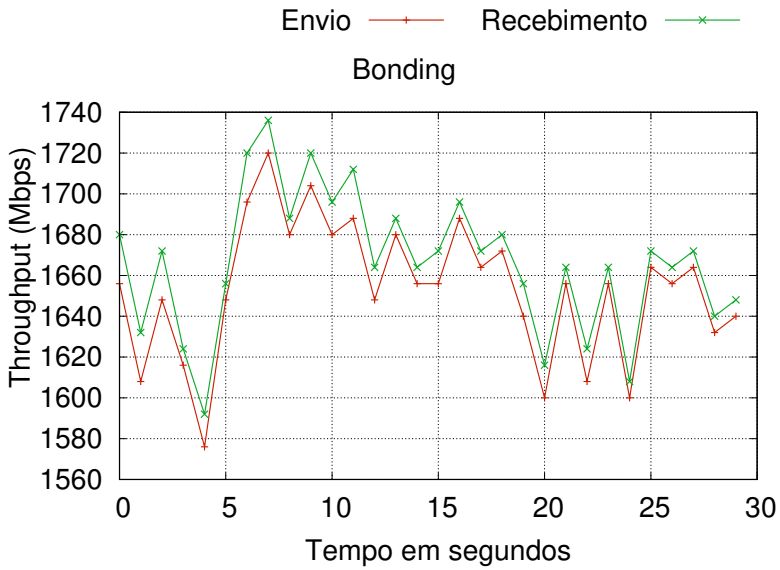


Com a configuração do *bonding* devidamente aplicada, passou-se a testar os demais modos. Inicialmente, aplicou-se o modo *Balance-rr*, o qual demonstrou correta funcionalidade, agregando as duas interfaces de rede e aumentando o *throughput* da rede e diminuindo a latência. Os demais modos, *active-backup*, *balance-xor*, *broadcast*, *802.3ad*, *balance-tlb* e *balance-alb* e *active-backup* e *broadcast* não foram avaliados e considerados neste trabalhos.

Passamos então a coleta de dados, inicialmente experimentou-se o *benchmark* Iperf3, segundo Iperf (2017), a utilização dos protocolos TCP e UDP pos-

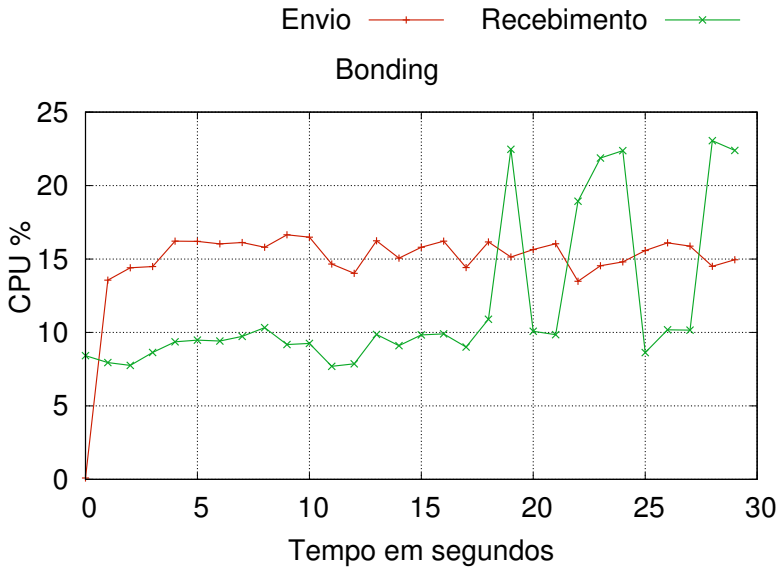
sibilitam a coleta do tráfego da rede, podendo então medir o *throughput* da rede, como demonstrado no Figura 3.4.

Figura 3.4: *Throughput* Iperf3



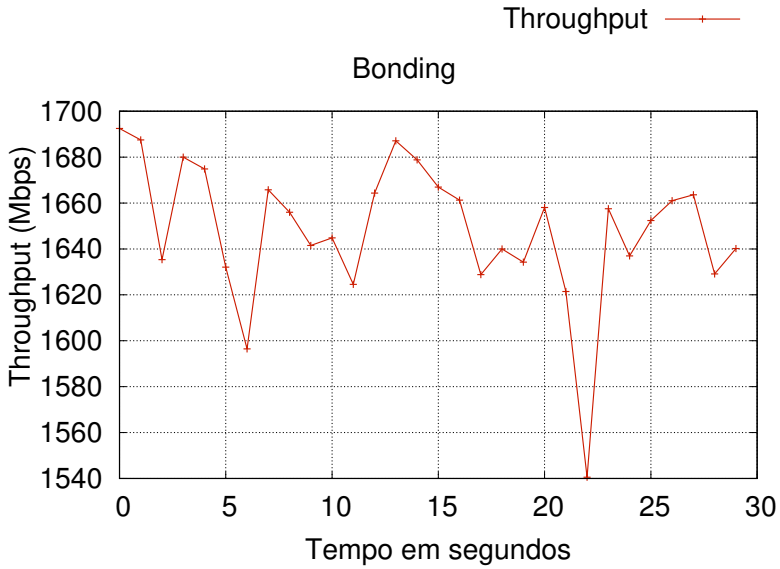
A Figura 3.4 trás informações do tráfego bidirecional, ou envio (*sender*) e recebimento (*receiver*), chegando á um máximo de 1740 Mbps para para o envio. Outra métrica coletada pelo *benchmark* foi a utilização de CPU, também coletada de forma bidirecional, chegando a um máximo de 16% para envio e 22% para recebimento conforme demonstrado na Figura 3.5. Lembrando que os testes foram executados 30 vezes.

Figura 3.5: Utilização da CPU



Estudando sobre a utilização do *benchmark* Iperf3, não encoutrou-se a forma de utilização para coleta de latência da rede, então excluindo a possibilidade de utilização deste *benchmark*.

Figura 3.6: *Throughput* NetPerf



Estudou-se a possibilidade da utilização do *benchmark* Netperf, na qual a coleta do *throughput* é possibilitada pela utilização de *sockets* TCP e UDP, entre *client* e *server*. A Figura 3.6 trás a informação do *throughput* da rede.

Pesquisou-se na documentação do *benchmark* NetPerf, porém também não encoutrou-se a forma correta de coletar a latência da rede, impossibilitando a utilização deste *benchmark*.

Utilizou-se então o *benchmark* NetPIPE para coleta dos resultados. Inicialmente, a escolha deste *benchmark* se deu por suprir todas as necessidades de avaliação e também pela facilidade de utilização, simplificando as coletas de resultados e formação dos gráficos.

Na Figura 3.7, encontra-se o *throughput* da rede, onde chegou-se a uma máxima de 1707 Mbps, conforme o aumento do tamanho dos pacotes. O *benchmark* NetPipe tem por característica alterar os tamanhos dos pacotes de forma aleatória, provendo interferência e oferecendo maior confiabilidade aos resultados.

Figura 3.7: *Throughput* NetPipe Ambiente Nativo

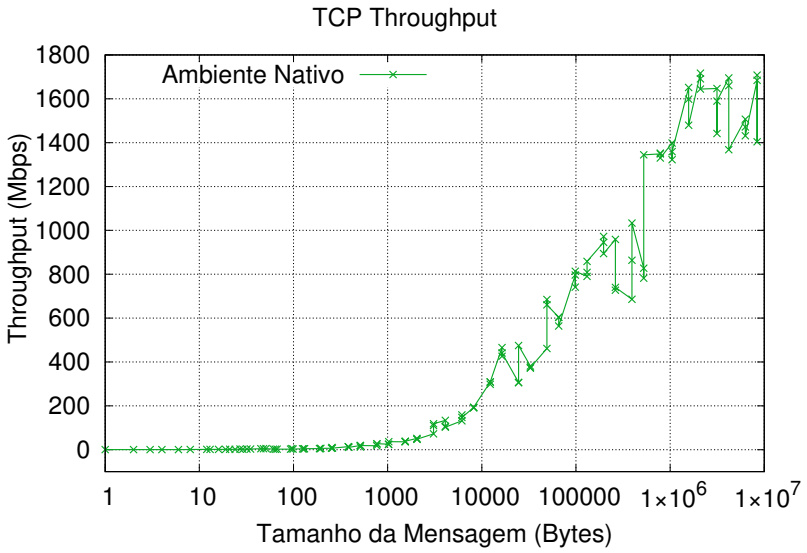
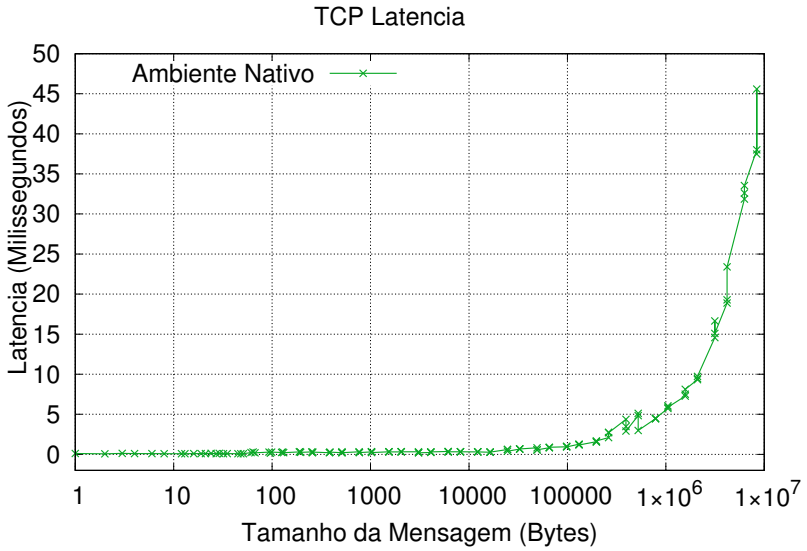


Figura 3.8: Latência NetPipe Ambiente Nativo



Fazendo uma análise mais profunda sobre o gráfico, houve pontos onde ocorreram congestionamentos. Segundo Köbel, Baluja García e Habermann (2012), isto ocorre devido ao funcionamento do algoritmo *Roundo-Robin*, o qual tem seu funcionamento baseado em tempos pré-definidos, nomeados como quantum.

Quando o tempo especificado pelo quantum finaliza, retornando de onde parou no próximo agendamento, este processo funciona de forma redundante, ocorrendo *overheads* vindos dos envios ainda não finalizados, o que ocasiona filas para o envio.

Outra métrica analisada foi a latência, a qual demonstrou uma redução significativa, chegando a 45 ms, como demonstrado na Figura 3.8.

3.2.1.3 Resultado Bonding aplicado em ambiente LXC

Para instalação do ambiente LXC, não pode-se utilizar da interface gráfica acessada por navegador, pelo fato do *host client* não estar com conexão a internet. As maiores implicações nas execuções das aplicações de LXC foram as dificuldades de conexão entre os contêineres. Isso pois para que haja comunicação, há a necessidade da interface ser utilizada pelo contêiner em modo *bridge* com a interface virtual utilizada pelo contêiner.

Mas pelo fato do protocolo *bonding* utilizar as interfaces físicas para comunicar-se através da interface *bond0*, o contêiner assume o *bond0* como uma interface *bridge*, não sendo possível a utilização desta interface por outra interface *bridge*. Em outras palavras, não podendo mesclar a interface *bridge* do contêiner à interface *bond0*.

A solução encontrada para contornar este problema, de acordo com Fisher (2016) acarreta em modificações no *kernel* do Linux, adicionando regras ao arquivo udev. A adição destas regras faz com que o sistema operacional do LXC identifique as interfaces físicas. Após realizadas as modificações no kernel, adicionou-se a interface *bridge* exigida para comunicação entre as interfaces, sendo necessário a adição do Código 3.3 no ambiente nativo.

Listing 3.3: Código Bridge

```
1 auto br0
2 iface br0 inet manual
3     bridge_ports bond0
4     bridge_fd 0
5     bridge_maxwait 0
```

Após esta configuração, se fez necessária a reinstalação do contêiner LXC, para que houvesse a detecção pela parte do contêiner da interface *bridge*.

Com o contêiner LXC corretamente instalado e os contêineres do *client* e do *server* comunicando-se, coletou-se os resultados através do *benchmark* NetPipe. A Figura 3.9 demonstra um resultado interessante, onde o *throughput* alcançou resultados similares ao ambiente nativo, obtendo uma máxima de 1742 Mbps. Podemos concluir que os *overheads* citados por Köbel, Baluja García e Habermann (2012) afetaram também o desempenho no ambiente LXC.

Figura 3.9: *Throughput* LXC.

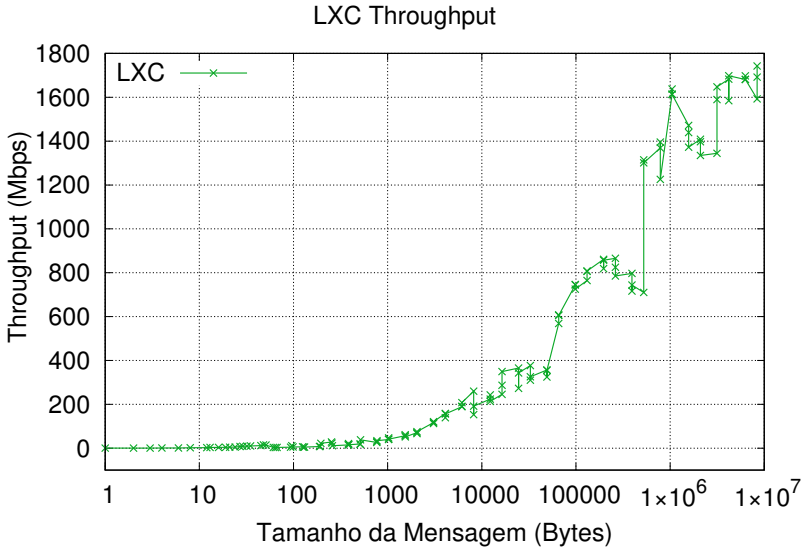
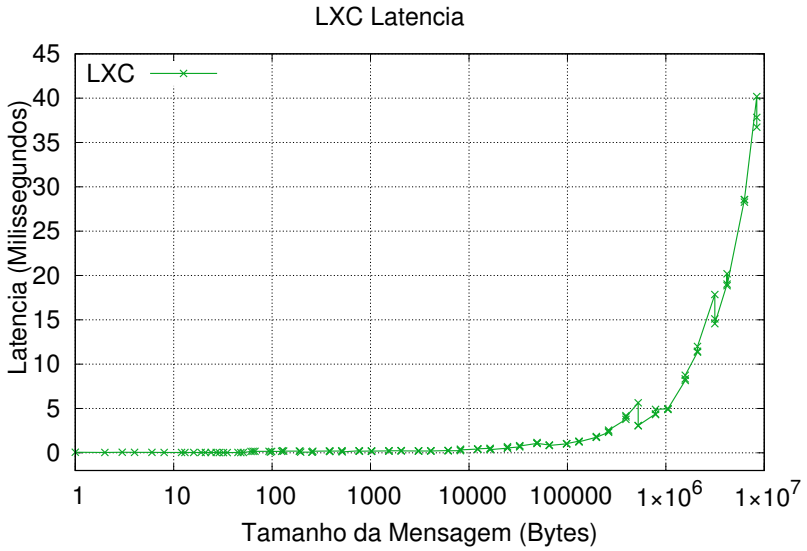


Figura 3.10: Latência LXC



Analisando a latência, conforme mostra o Figura 3.10, os resultados também aproximaram-se muito ao ambiente nativo, chegando a latência de 40 ms.

3.2.1.4 Resultado Bonding aplicado em ambiente KVM

Para a instalação do KVM necessitou-se da instalação de mais uma interface no *client*, para prover a conexão com a interface gráfica para a configuração da rede, sendo necessária para direcionar o tráfego pela interface *bond0*. Com a comunicação entre as máquinas virtuais estabelecidas, realizou-se a coleta do tráfego.

Figura 3.11: *Throughput* KVM.

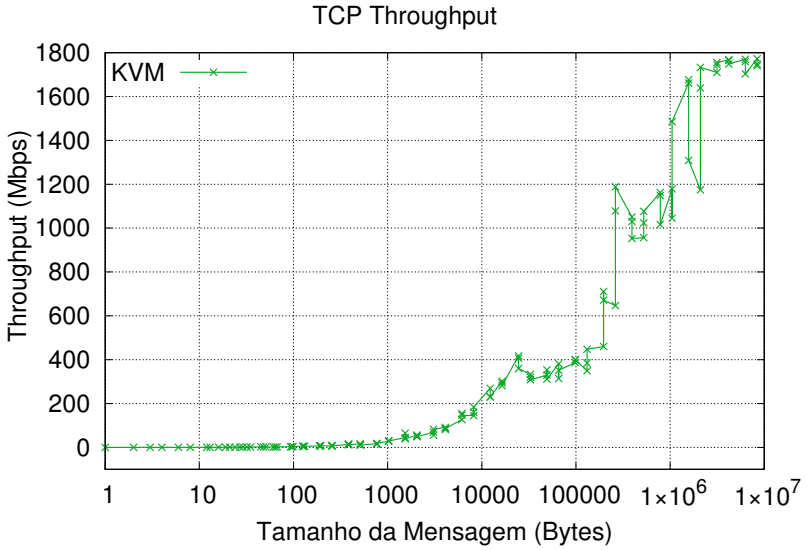
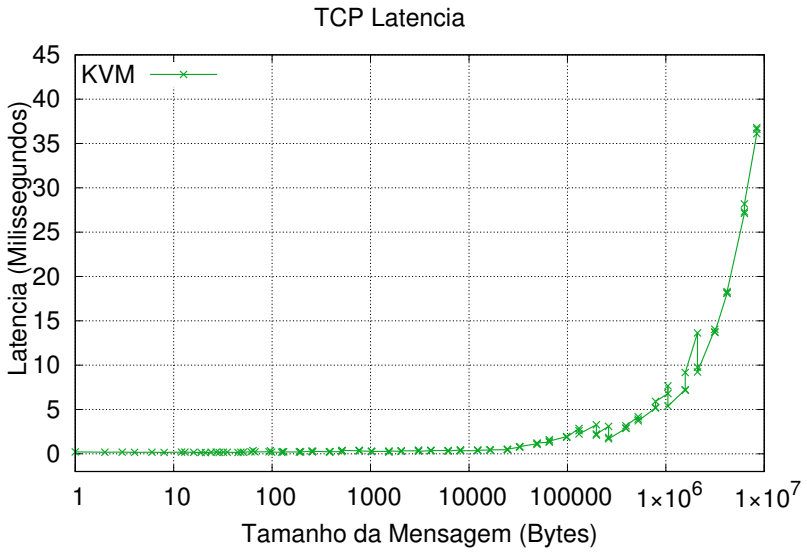


Figura 3.12: Latência KVM



Analisando os resultados, também houve muita semelhança ao ambiente nativo, chegando a uma máxima de 1748 Mpbs e latência de 36 ms, conforme nas

Figuras 3.11 e 3.12.

3.2.1.5 Comparação entre os resultados bonding para Ambiente Nativo, LXC e KVM

A comparação dos resultados se inicia com a Figura 3.13, a qual contém o *throughput* dos ambientes nativos, KVM e LXC, onde houve algo interessante, todos os ambientes alcançaram aproximadamente o mesmo *throughtput*.

Notou-se uma queda brusca no desempenho a partir de 400Mbps de *throughput*, resultantes a contadores TCP. Segundo Matteussi et al. (2014), há gargalos ocorridos aos parâmetros *Socket Buffers* (rmem_size, wmen_size) e *Transmit Queue Lenght* (txqueuelen).

Houve diferenças em relação ao KVM, onde ocorreu um *overhead* de tráfego para tamanho de pacotes de 100000 *Bytes*. Segundo Trindade e Costa (2018), se dá ao fato do KVM tratar os dados das interrupções da arquitetura dependente da virtualização baseada em *hardware* possuir sobrecarga em relação a troca de processos entre o KVM e o *host* nativo. Assim, ocasionando números elevados de ciclos de processamento.

Relacionado a latência, a Figura 3.14 demonstra um resulatdo interessante, onde houve redução da latência com instâncias LXC e KVM.

3.2.2 Ambiente de testes *Multipath TCP*

Para o ambiente de testes, utilizou-se duas máquinas no ambiente nativo, dispondo de três interfaces de placa de rede. A placa principal é desativa na realização dos testes, as outras duas placas são configuradas e usadas como escravas. Logo após, instalou-se e configurou-se o *Multipath* TCP. O sistema operacional adotado nas máquinas foi o Linux Ubuntu Server 16.04.

Ambas as máquinas tem o mesmo objetivo, utilizar as configurações do

Figura 3.13: *Throughput* Ambiente Nativo, LXC e KVM

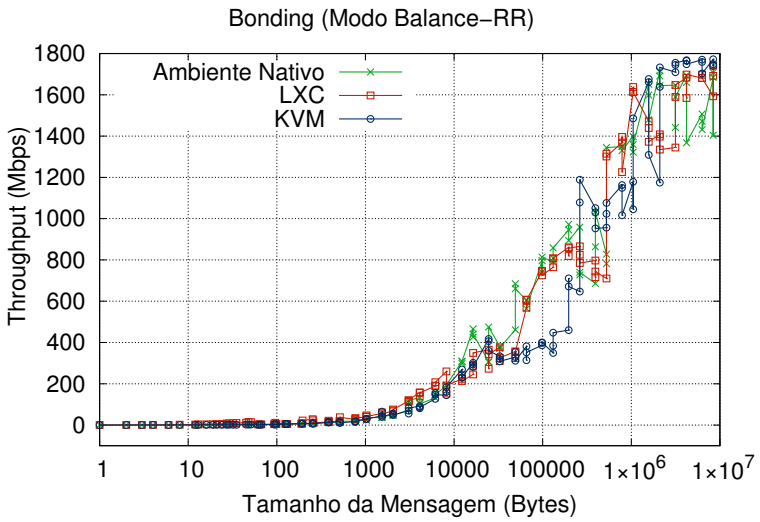
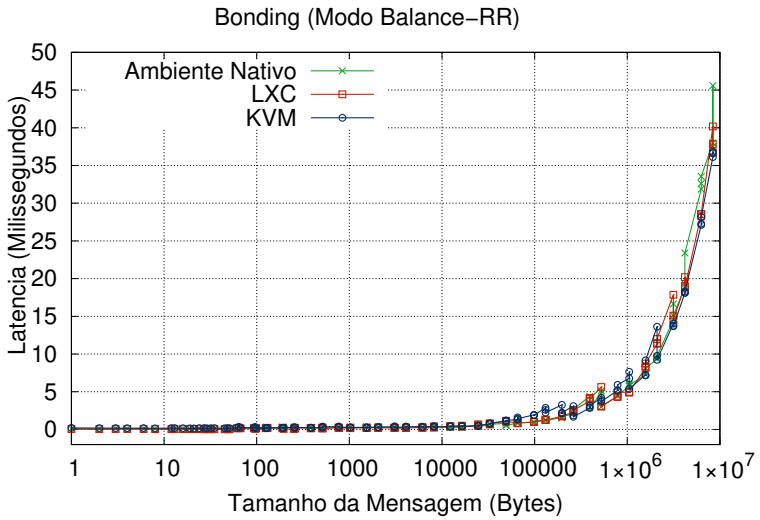


Figura 3.14: Latência Ambiente Nativo, LXC e KVM



Multipath TCP para a troca de informações entre elas, oferecendo agilidade, eficiência e confiança sobre a comunicação para gerar os resultados esperados.

A ideia principal da utilização do *Multipath* TCP é o aumento do *throughput* entre duas máquinas. O MPTCP cria vários fluxos ou caminhos para os dados serem enviados, assim gerando rapidez e eficiência na comunicação. Quando um dos caminhos estiver em conflito ou bloqueado, a comunicação não vai utilizar a rota que esta com problema, seguindo outros caminhos disponíveis.

O *Hardware* da máquina, no qual foi instalado o *Multipath*, necessitam de

Quadro 3.2: Hardware

	Processador	Placas de Rede	Memória RAM
Servidor	Intel(R) Core(TM) i5 CPU, 3.20ghz	3 Interfaces Gigabit	4GB
Cliente	Intel(R) Core(TM) i5 CPU, 3.20ghz	3 Interfaces Gigabit	4GB

especificações e configurações mais atuais para não ter problemas com lentidão ou falhas quando é instalado ou configurado o virtualizador. Para comunicação entre as placas, foi utilizado os cabos de rede CAT6. No Quadro 3.2 citado acima, segue a configuração dos hardwares de cada *host*.

Como pode ser visto no Quadro 3.2, os dois *host* possuem configurações de *hardware* iguais. Foram usadas quatro placas de rede para os testes do *Multi-path* e uma placa de rede principal do *host* para acessar a Internet. Para o trabalho, foi usado o modo padrão do MPTCP, já o *Rond Robin* e *Redundant* não foram avaliados e considerados neste trabalho.

3.2.2.1 Testando Transmission Control Protocol - TCP

O protocolo TCP é conhecido por entregar com segurança e integridade a sequência de dados que são transmitidos na rede. Foi criado para que máquinas distintas pudessem estabelecer conexão entre as mesmas, o que acabou alavancando a internet. Embora o TCP tenha sido definido em 1981, ele continua sendo amplamente utilizado nos dias atuais. É composto por 4 camadas, sendo elas; aplicação, transporte, rede e interface, Wetherall e Tanenbaum (2011)

Os testes demonstrados na Figura 3.15 e 3.16 são referentes ao TCP. Como pode ser observado, o TCP trabalha com uma placa de rede. A placa tem capacidade de 1000 Mbps, e de acordo com o resultado do teste, a comunicação é de 894 Mbps e a latência da rede foi de 74ms. Utilizou-se para comunicação duas máquinas, ambas com os mesmo sistema operacional, configurações de rede e *Hardware* iguais, assim gerando os resultados demonstrados na Figura 3.15 e 3.16.

Figura 3.15: Throughput TCP

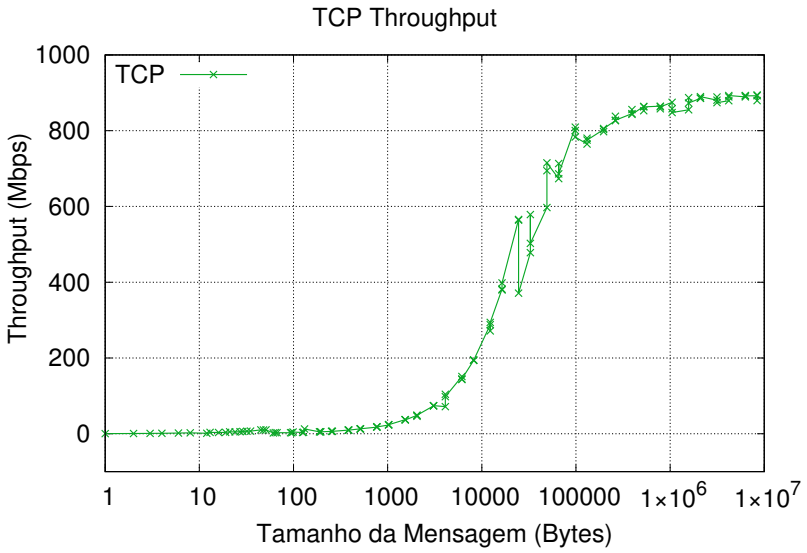
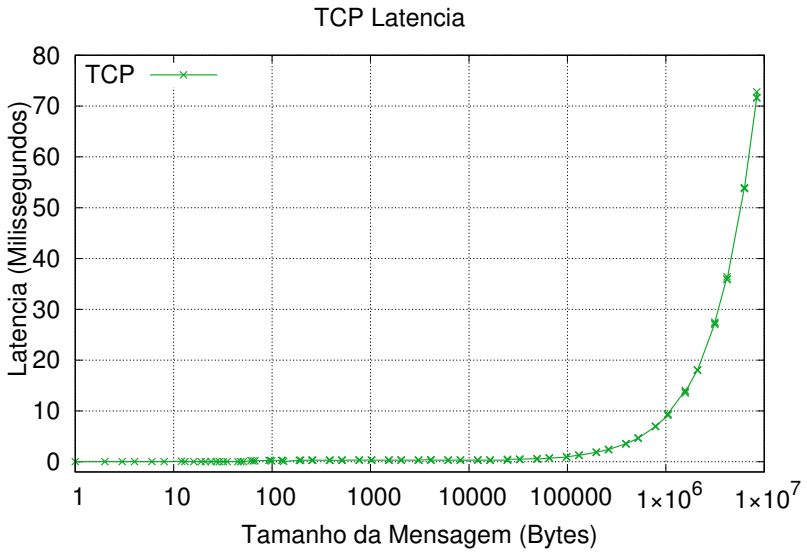


Figura 3.16: Latencia TCP



Listing 3.4: Configuração Interface TCP

```
# source /etc/network/interfaces
#The loopeback network interface
auto lo
iface lo inet loopback
# enp0s25
iface enp0s25 inet static
address 192.168.2.27
netmask 255.255.255.0
gateway 192.168.2.1
```

Na Figura 3.4 podemos ver a atribuição de IPs de um dos *host* e na Figura 3.5 tem os IPs atribuídos para outro *host*.

Listing 3.5: Configuração Interface TCP

```
# source /etc/network/interfaces
#The loopeback network interface
auto lo
iface lo inet loopback
# enp0s25
iface enp0s25 inet static
address 192.168.2.15
netmask 255.255.255.0
gateway 192.168.2.1
```

Como pode ser analisado na Figura 3.5, atribuiu-se um IP com o endereço de `192.168.2.15` para a interface principal. Para geração dos resultados, utilizou-se o *Benchmark* Netpipe instalado em ambas as máquinas. Assim, coletando o *Throughput* e latência da rede como são demonstrados nas Figuras 3.15 e 3.16.

3.2.2.2 Configuração Multipath TCP

O *Multipath* TCP permite que uma conexão use vários caminhos para maximizar o uso de recursos da rede e aumentar a redundância. A redundância oferecida pelo *Multipath*, utiliza todos os canais que estão disponíveis na rede para aumentar o *throughput*, não utilizando somente um canal, e desta forma o *multipath* trás benefícios de desempenho.

E ainda referente aos benefícios do *Multipath* TCP, um deles é que em caso de falha entre uma das interfaces de rede, uma conexão MPTCP pode continuar ainda ativa, assim não perdendo pacotes que estão trafegando na comunicação dos *host*.

Na Figura 3.6 podemos analisar os IPs atribuídos para o MPTCP, adicionando novas placas de rede para realizar os testes.

Listing 3.6: Configuração Interface Host 1

```
1
2  # The loopback network interface
3  auto lo
4  iface lo inet loopback
5
6  # The primary network interface
7  auto enp0s25
8  iface enp0s25 inet manual
9
10 auto eth0
11 iface eth0 inet static
12 bridge_ports enp0s25
13 bridge_stp off
14 bridge_fd 9
15 address 192.168.2.27
16 netmask 255.255.255.0
17 gateway 192.168.2.1
18 dns-nameservers 8.8.8.8
19
20 #ens128
21 auto ens128
22 iface ens128 inet manual
23 ##Bridge  Name ###
24 auto eth1
25 ### Bridge Information
26 iface eth1 inet static
27 bridge_ports ens128
28 bridge_stp off
29 bridge_fd 9
30 ### Bridge IP ###
31 address 10.1.1.2
32 network 10.1.1.0
33 netmask 255.255.255.0
34 gateway 10.1.1.1
35 dns-nameservers 8.8.8.8
36
37 #ens132
38 auto ens132
```

```
39 iface ens132 inet manual
40 ##Bridge  Name ###
41 auto eth2
42 ### Bridge Information
43 iface eth2 inet static
44 bridge_ports ens132
45 bridge_stp off
46 bridge_fd 9
47 ### Bridge IP ###
48 address 10.1.2.2
49 network 10.1.2.0
50 netmask 255.255.255.0
51 gateway 10.1.2.1
52 dns-nameservers 8.8.8.8
```

Inicialmente, antes das configurações do *Multipath* TCP, atribuiu-se IPs as placas do *host* 1, a interface `enp0s25` com IP `192.168.2.27` utiliza-se para acesso a internet. Já as outras duas placas, são as placas que foram adicionadas para realizar a agregação entre elas e melhorar a comunicação entre os *host*. Na interface `ens128` foi atribuído o IP `10.1.1.2` e a interface de rede com o nome `ens132` atribuído o IP `10.1.2.2`.

Na Figura 3.7, mostramos as configurações do *host* 2.

Listing 3.7: Configuração Interface Host 2

```
1
2 # The loopback network interface
3 auto lo
4 iface lo inet loopback
5
6 # The primary network interface
7 auto enp0s25
8 iface enp0s25 inet manual
9 #       address 192.168.2.15
10 #       netmask 255.255.255.0
11 #       gateway 192.168.2.1
12
```

```
##Bridge  Name ###
auto eth0
### Bridge Information
iface eth0 inet static
bridge_ports enp0s25
bridge_stp off
bridge_fd 9
### Bridge IP ###
address 192.168.2.15
netmask 255.255.255.0
gateway 192.168.2.1
dns-nameservers 8.8.8.8

#ens128
auto ens128
iface ens128 inet manual
 #       address 10.1.1.3
  #      netmask 255.255.255.0
   #     gateway 10.1.1.1

##Bridge  Name ###
auto eth1
### Bridge Information
iface eth1 inet static
bridge_ports ens128
bridge_stp off
```

Este *host* utilizará 3 interfaces. Na placa principal com a interface `enp0s25` utiliza o IP `192.168.2.15`, e as outras duas placas que serão usadas para os testes, a `ens128 10.1.1.3` e a `ens132 10.1.2.3`.

Criou-se uma rede agregada para a comunicação entre as quatro placas que estão configuradas, isso se chama de ponte ou *bridge*, que permite que a comunicação aconteça entre as quatro placas de rede, não permitindo que as informações saiam deste laço.

As placas de rede adicionais foram configuradas para realizar a agregação

de *link*, no qual são adicionadas rotas a serem seguidas para que haja comunicação entre as máquinas. Na Figura 3.8 esta representando as placas de redes disponíveis no computador.

Listing 3.8: Placas de rede

```
# ip rule show
# Host1
from all lookup local
from 10.1.2.2 lookup 2
from 10.1.1.2 lookup 1
from all lookup main
from all lookup default

# Host2
from all lookup local
from 10.1.2.3 lookup 2
from 10.1.1.3 lookup 1
from all lookup main
from all lookup default
```

Depois de configurado e realizado atualização do sistema, a placa principal que usa internet é desativada, e as outras duas placas que sobraram de cada *host*, ficam para fazer os testes de agregação de *link*.

Depois de várias tentativas e testes para o *Multipath* TCP, alcançou-se os resultados esperados, que foram agregar as duas placas de rede e chegar a um resultado de 1.800 Mbps, como mostrado nas Figuras 3.17 e 3.18.

Para realizar o teste do *Multipath* TCP, utilizou-se o *Benchmark* Netpipe. As placas que foram usadas e configuradas foram com margem de IPs `10.1.1.2` e `10.1.2.1`. Os resultados foram positivos, agregando as duas placas de rede formando só uma, o resultado do *throughput* foi de 1742 Mbps e latência foi de 38,039ms. Analisando a imagem, houve uma pequena queda quando o resultado chegou aos 100000 Bytes, mas logo em seguida manteve seus resultados, terminando o teste com o MPTCP. Segundo Nguyen e Nguyen (2011), o MPTCP através

Figura 3.17: Throughput MPTCP

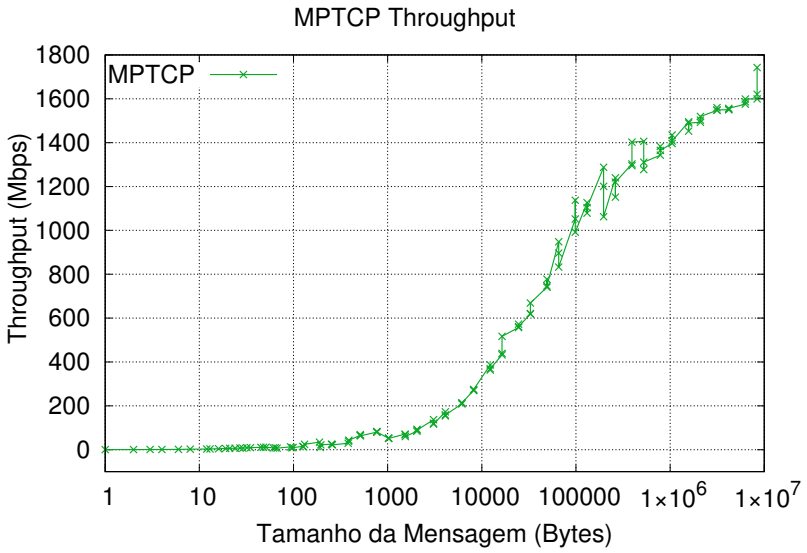
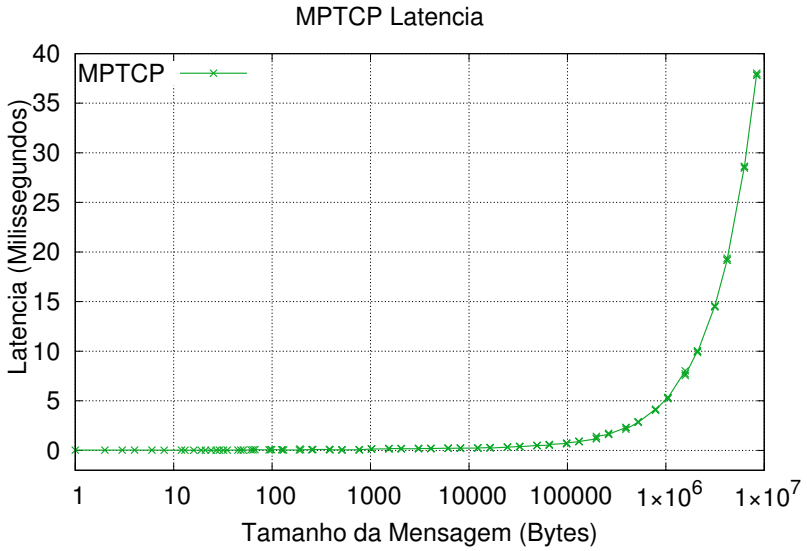


Figura 3.18: Latência MPTCP



de vários caminhos deve atingir pelo menos uma taxa de transferência melhor que única conexão TCP de 800 Mbps. O MPTCP como controle na rede, proporciona um melhor desempenho, agregando as placas de rede formando somente uma, deixando as placas com capacidade maiores, chegando a 2000 Mbps.

Com os testes realizados até agora, comprovou-se que o *Multipath* TCP é mais eficiente se comparado com o TCP, os resultados praticamente duplicaram, como podemos analisar os testes que foram gerados.

3.2.3 Configuração Linux Containers (LXC)

O LXC é um *Container* e utiliza o sistema operacional do ambiente nativo. Mesmo o LXC sendo hospedeiro, nele pode ser adicionado configurações de rede diferentes, pelo beneficio de poder ter *IPs* diferentes e, oferecendo um ambiente mais próximo de um ambiente nativo.

Figura 3.19: Throughput LXC

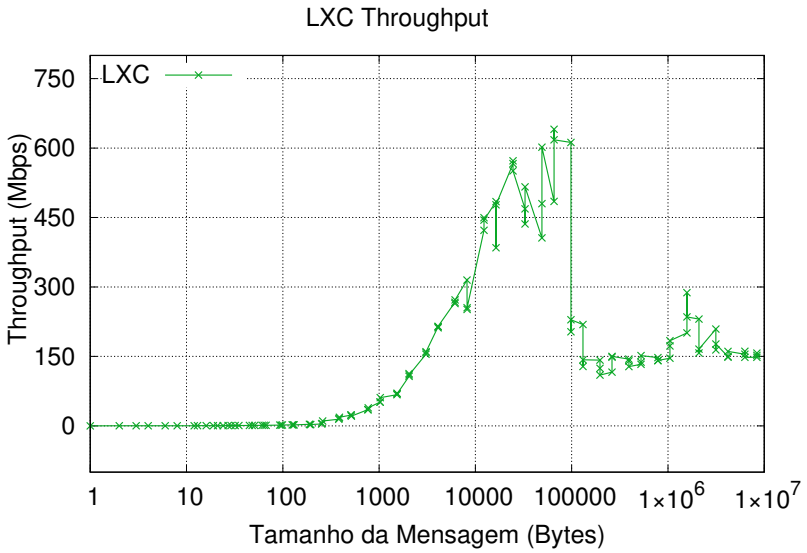
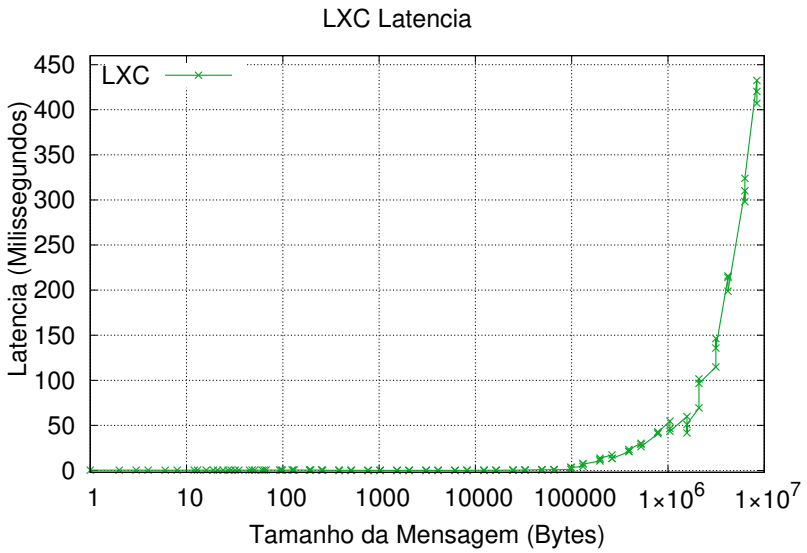


Figura 3.20: Latência LXC



Analisando os resultados da Figura 3.19, pode-se perceber que a agregação não foi bem sucedida, uma vez que os resultados não ultrapassaram o máximo throughput teórico de uma placa de rede. Quando o testes chegaram aos 100000

Bytes aconteceu uma perda brusca no desempenho. A partir desse ponto o resultado manteve-se com Throughput baixo oscilando entre 150 a 300 Mbps.

Como os testes foram realizados para o MPTCP, utilizou-se as mesmas configurações para gerar os testes do LXC, mas não teve-se o mesmo resultado positivo quanto ao MPTCP conforme demonstrado nas Figuras 3.19 e 3.20. Como pode-se observar, este resultado não foi o esperado, as configurações foram as mesmas nesta plataforma, mas para este teste o resultado foi muito inferior.

Os contêineres criados, não conseguiam se comunicar, sendo necessário fazer mudanças no *Kernel* do sistema operacional. As mudanças que são necessárias não chegaram ao nosso grau de estudo e conhecimento, não conseguindo modificar ou fazer as alterações. Sendo assim, os resultados foram negativos. Segundo Qiu (2016), diz que para a comunicação dos contêineres junto com o MPTCP, precisa de softwares adicionais para ter a comunicação e adicionar outras rotas com as placas agregadas.

3.2.4 Kernel-based Virtual Machine (KVM)

O KVM cria máquinas virtuais que emulam e usam componentes de *Hardware* e *software*, tornando assim uma prática útil e eficaz, sem ter a necessidade de ter plataformas adicionais.

Foram configuradas os dois *hosts* que usam o virtualizador KVM com o MPTCP, mas com IPs diferentes e rotas diferente. Abaixo podemos analisar os resultados do teste.

Analisando a implementação do MPTCP no Ambiente Nativo, realizou-se a instalação da agregação de *link* no virtualizador KVM. Utilizou-se a mesma configuração dos testes anteriores, apenas realizando modifições de IP. Observando a Figura 3.21, o uso do KVM com MPTCP tem resultado semelhante ao do ambiente nativo, chegando a 1800 Mbps. Conforme Teka, Lung e Ajila (2015), os resultados experimentais demonstram a viabilidade e melhoria do desempenho da rede

Figura 3.21: Throughput KVM

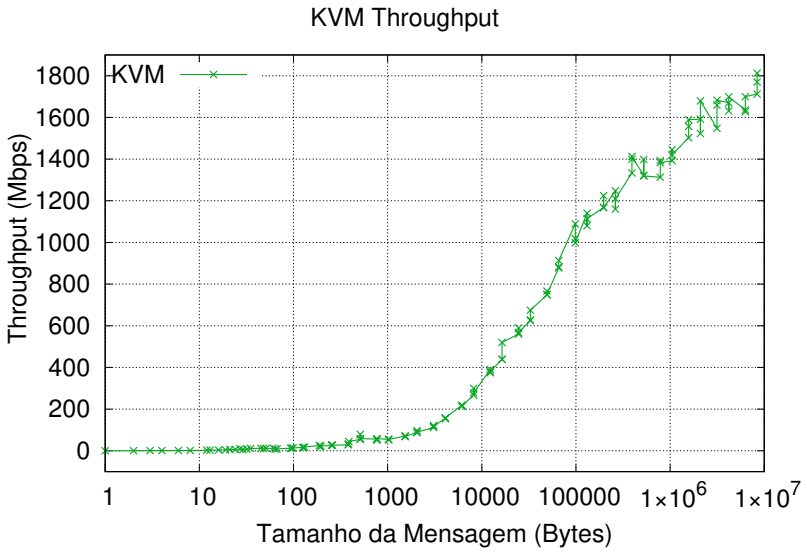
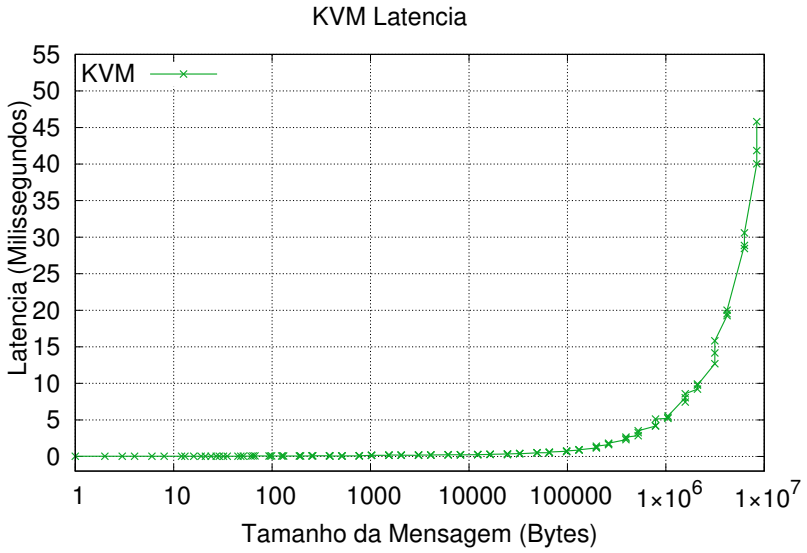


Figura 3.22: Latência KVM



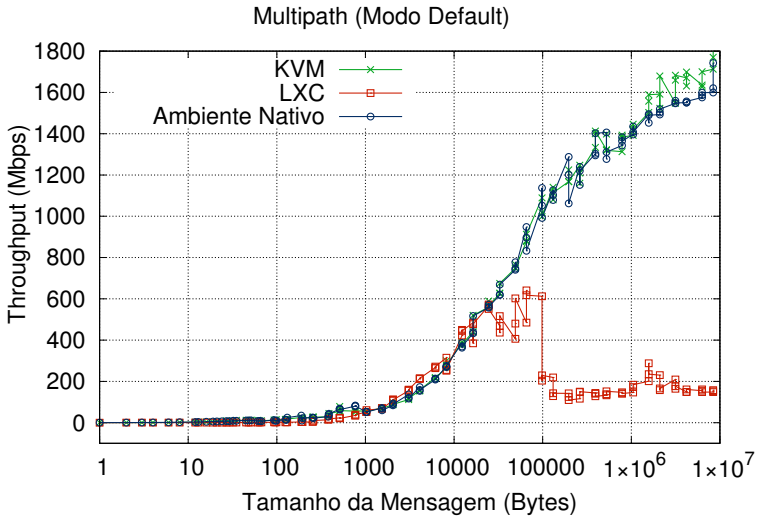
usando o KVM, sujeito a melhorias nas tomadas de decisões para escolher a rota mais rápida e confiável.

Com os testes realizados até agora, comprovou-se que o *Multipath* TCP usado no KVM também consegue fazer a agregação de *link* e ter resultados positivos para a rede.

3.2.4.1 Resultados Multipath TCP para Ambiente Nativo, LXC e KVM

De ambos os testes realizados sobre a agregação de *link*, abaixo temos uma comparação de ambientes de virtualização em Ambiente Nativo *Multipath*, LXC e KVM, fazendo comparações de *throughput* e *latência*.

Figura 3.23: *Throughput* Ambiente Nativo, LXC e KVM

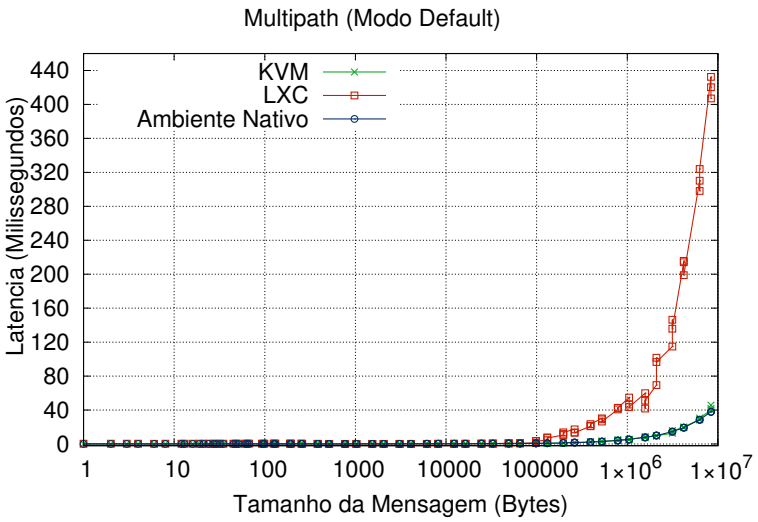


Como pode ser analizado na Figura 3.23 em relação ao *throughput*, os resultados do MPTCP nativo e o KVM foram similares, a diferença que começa aparecer é quando o resultado do KVM esta próximo aos 1000000 Bytes, mas ambos os resultados chegam a 1795 Mbps no resultado final. Sobre o LXC, a agregação não ocorreu como o esperado, onde houveram problemas na virtualização dos contêineres, não possibilitando a gerencia das placas de rede para realizar a agregação e gerar os resultados.

Como um dos objetivos, duas das agregações dos resultados foram positivo, o MPTCP e KVM, conseguindo fazer a agregação das placas e gerar resultados. Assim, mostramos como é possível a agregação de *link* usando ambientes de virtualização diferente, como MPTCP e KVM. Na Figura 3.24 nota-se o resultado da latência, mostrando quanto tempo leva para um pacote de dados ir de um ponto para o outro.

Com o resultado demonstrado na Figura 3.24, pode-se ver que o *Multipath*

Figura 3.24: Latência Ambiente Nativo, LXC e KVM



Nativo e o *KVM* alcançaram os resultados esperados, chegando a um tempo de 40 milissegundos para enviar os dados de uma destino até o outro. O LXC como não teve resultados positivos no *throughput*, sendo o resultado fora do esperado.

3.2.5 Dificuldades Encontradas

3.2.5.1 MPTPC

Em geral, obteve-se resultados positivos, porém, ocorreram dificuldades para a concretização dos resultados. Com o *Multipath TCP* padrão, implatou-se a aplicação MPTCP nas duas máquinas onde foram realizado os testes, ambas as máquinas comunicando-se e realizando a agregação, assim gerando os resultados esperados.

Na outra etapa, a agregação foi realizada com o KVM. As interfaces de rede foram copiadas do teste anterior *Multipath* TCP. Os primeiros problemas ocorreram com a comunicação entre as duas máquinas que estavam sendo virtualizadas, no qual não houve comunicação, não aplicando a agregação de *link*. Analisando diversos artigos, e realizando vários novos testes, concretizou-se resultados positivos, ocorrendo comunicação entre as máquinas e gerando os resultados esperados.

Sobre o último teste da aplicação do LXC, não alcançou-se o resultado esperado. Inicialmente ocorreram problemas com aplicações dos contêineres, não havendo permissão para inicialização. Estudou-se então artigos para resolver este problema, obteve-se bom proveito sobre o estudo, conseguindo então iniciar os ambientes de testes. Logo depois, realizou-se as implementações do MPTCP em ambos os ambientes, e aplicando os testes. Os testes foram de forma negativa comparando com os testes das agregações de Ambiente Nativo e o KVM. Voltando aos estudos para resolver o problema, e por fim encontrou-se artigos Qiu (2016) que mostrando a necessidade de realizar alterações no *Kernel* do sistema operacional, ou usar outra ferramenta de virtualização para conseguir realizar os testes desejados. Por conta destes fatores não puderam ser realizados os testes e também pela dificuldade de reconfiguração do kernel, sendo assim não gerando resultado positivo ao teste de LXC.

3.2.5.2 Bonding

As dificuldades enfrentadas em relação ao protocolo *bonding* ocorrem inicialmente na configuração do protocolo, o qual apresenta divergências relacionadas a compatibilidade relacionadas ao sistema operacional. Realizou-se diversos testes até a correta comunicação entre os *hosts*.

Outro problema ocorreu referente a aplicação dos modos de utilização do protocolo *bonding*, onde somente o modo *Balance-rr* demonstrou correta funcionalidade agregando as interfaces físicas em um único *link* lógico. Segundo Bright (2017), o *driver* deve ter suporte *Ethtool* para recuperar a velocidade de cada escravo, mas mesmo havendo suporte as interfaces não funcionaram de maneira correta.

Em relação a instalação do ambiente LXC, houve dificuldade relacionadas a comunicação entre as interfaces *bridge* necessárias para a comunicação entre contêineres. Isso se deve pelo fato da interface lógica *bond0* ser interpretada pelo sistema operacional como uma interface *bridge*, não havendo a possibilidade do

contêiner LXC utilizar a interface *bond0*.

Para contornar este problema, houve a necessidade de modificações no *kernel* do Linux, onde, segundo Fisher (2016), devem ser adicionas regras ao arquivo 'udev', possibilitando ao sistema operacional identificar as interfaces físicas.

3.3 COMPROVAÇÃO DE HIPÓTESES

Foram apresentados nas seções anteriores diversos gráficos, os quais mostraram o comportamento dos protocolos utilizados nesta pesquisa, demonstrando os resultados no ambiente nativo, LXC e KVM.

3.3.1 Primeira Hipótese

- A vazão de rede usando MPTCP e *Bonding* diverge significativamente em instâncias de ambiente de nuvem baseado em instâncias LXC.

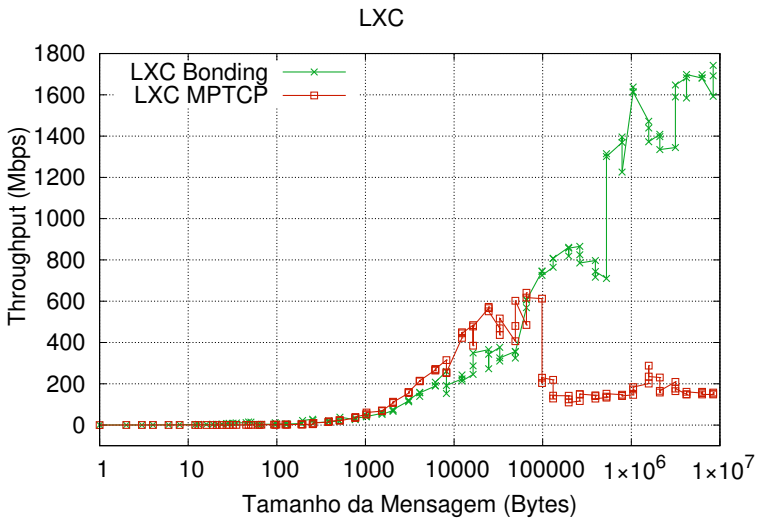
A primeira hipótese foi aceita, pelo fato de somente o protocolo *bonding* funcionar corretamente em instâncias de ambiente de nuvem baseados em instâncias LXC. Na Figura 3.25 pode-se notar a diferença de funcionamento nos dois protocolos. O protocolo *bonding* agregou as interfaces de rede, chegando á 1720 Mbps enquanto o protocolo MPTCP não agregou as interfaces de rede, ainda ocorrendo *overheads* no desempenho, chegando a somente 630 Mbps.

3.3.2 Segunda Hipótese

- A vazão de rede usando *MPTCP* e *Bonding* diverge significativamente em instâncias de ambiente de nuvem baseado em instâncias KVM.

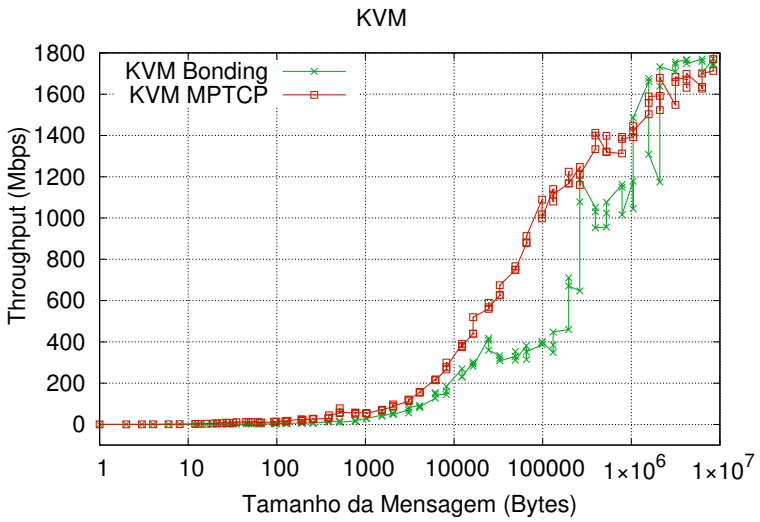
A segunda hipótese foi corroborada pelo fato dos protocolos *bonding* e MPTCP não divergirem significativamente em instâncias de ambiente de nuvem baseado em instâncias KVM. Conforme demostrado na Figura 3.26, nota-se o que

Figura 3.25: *Throughput Bonding* x MPTCP



ambos os protocolos alcançaram um *throughput* máximo próximo a 1800 Mbps. As diferenças encontram-se principalmente em relação ao protocolo *bonding*, no qual houve *overheads* entre tamanhos de pacotes de 10000 *bytes* a 1000000 *bytes*.

Figura 3.26: *Throughput Bonding* x KVM



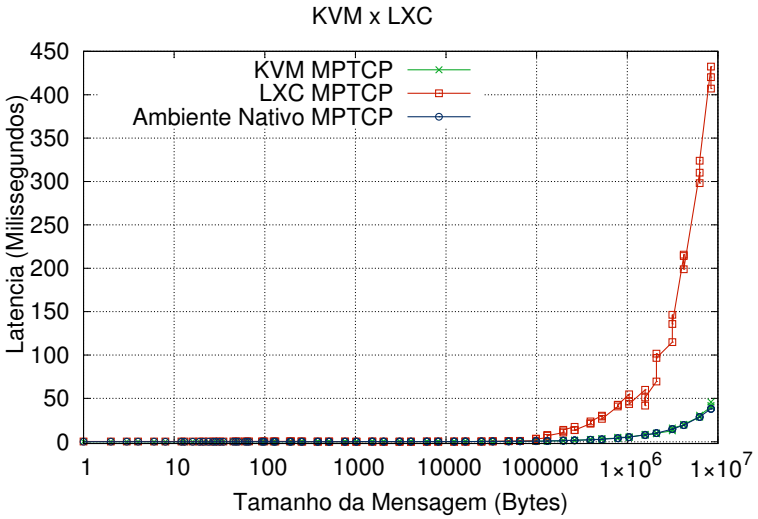
3.3.3 Terceira Hipótese

- A latência da rede dos protocolos *MPTCP* e *Bonding* aumenta significativamente ao implantá-los em ambientes de nuvem com instâncias LXC e KVM.

Para a terceira hipótese, comprovou-se o aumento da latência quando apli-

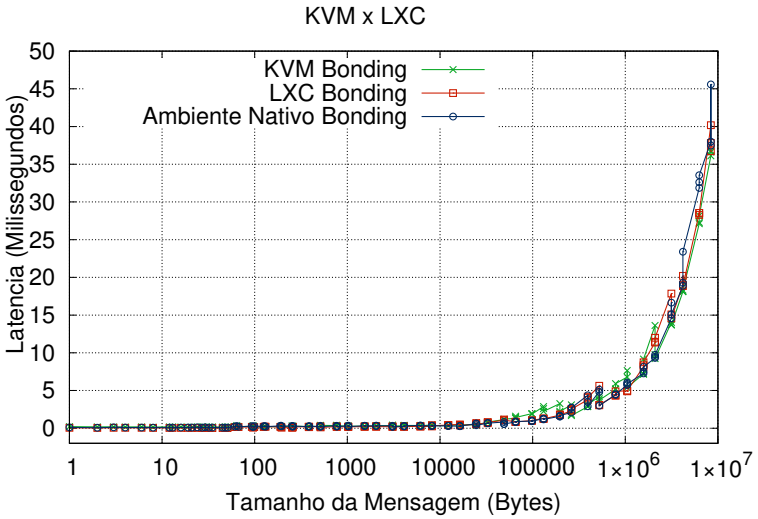
cado o protocolo MPTCP em ambientes de nuvem com instancias LXC e KVM em comparação ao ambiente nativo atuando com o protocolo MPTCP, conforme demostrados nas Figuras 3.27 . Assim corroborando a terceira hipótese.

Figura 3.27: Latência Ambiente Nativo x LXC x KVM



Em relação ao protocolo *bonding*, a latência em ambientes de nuvem com instâncias LXC e KVM reduziu em 5 milissegundo para LXC e 9 milissegundos para KVM, como demonstrado na Figura 3.28.

Figura 3.28: Latência Ambiente Nativo x LXC x KVM



CONCLUSÃO

A realização da pesquisa feita por este trabalho buscou avaliar o desempenho do protocolo *bonding* e MPTCP em instâncias LXC e KVM em comparação ao ambiente nativo. Este trabalho propiciou a aprendizagem de ferramentas como o LXC e KVM, bem como a configuração e execução dos protocolos *bonding* e MPTCP, além da realização de testes com *benchmarks* específicos de análise do ambiente de rede, o que possibilitou a avaliação dos componentes utilizados.

O problema deste trabalho foi descobrir quais dos protocolos de agregação de *link* oferece o melhor desempenho em um ambiente de nuvem privada com instâncias LXC e KVM?

Levando em consideração as hipóteses levantadas para responder ao problema de pesquisa, as quais incentivaram este trabalho, a primeira afirma que há divergências significativas ao aplicar os protocolos MPTCP e *bonding* em ambiente de nuvem baseados em instâncias LXC em relação a vazão da rede. Esta hipótese foi comprovada, pois como mostrado na Figura 3.25, há divergências negativas em relação ao protocolo MPTCP, o qual demonstrou perdas de desempenho no *throughput*. Isto se deu pelo fato de haver a necessidade de realizar alterações no *kernel* do Linux, conforme Qiu (2016).

Já o protocolo *bonding* praticamente não houve diferença em relação ao ambiente nativo.

A segunda hipótese, afirma que há divergências significativas ao aplicar os protocolos MPTCP e *bonding* em ambiente de nuvem baseados em instâncias KVM em relação a vazão da rede. Diferentemente da primeira hipótese, a segunda hipótese foi corroborada pelo fato do protocolo MPTCP não ter alcançado os resultados esperados, conforme demonstrado na Figura 3.26. Em relação ao protocolo *bonding*, houveram divergência, porém não significativas, resultantes aos contadores TCP, segundo Matteussi et al. (2014), ocorrem gargalos ocorridos aos parâmetros *Socket Buffers* (rmem_size, wmen_size) e *Transmit Queue Lenght* (txqueuelen).

Para a terceira hipótese, a qual afirma que há divergências significativas ao aplicar os protocolos MPTCP e *bonding* em ambientes de nuvem com instâncias LXC e KVM em relação a latência da rede, houve divergências por parte do protocolo MPTCP, no qual houve aumento com instâncias LXC, conforme a Figura 3.26. E para o protocolo *bonding* os resultados foram praticamente iguais ao ambiente nativo, não havendo divergências entre o ambiente nativo e com instâncias LXC e KVM, como monstrado na Figura 3.28.

O principal objetivo deste trabalho foi alcançado, o qual buscou-se avaliar e comparar o desempenho dos protocolos *Bonding* e *MPTCP* em ambientes de computação em nuvem.

As contribuições deste trabalho foram principalmente em demonstrar o desempenho do protocolo *bonding* na agregação das interfaces físicas aumentando o *throughput* e diminuindo a latência da rede. Ao aplicar o protocolo *bonding* e em instâncias LXC e KVM o desempenho da rede aproxima-se ao ambiente nativo.

Outra contribuição está na utilização do protocolo MPTCP, onde demonstrou-se estabilidade na agregação de *links*, e quando aplicado em instâncias KVM, o desempenho aproxima-se do ambiente nativo.

Com os resultados coletados, pretende-se realizar a escrita de artigos com o conteúdo de nossa pesquisa. Primeiramente, planeja-se aprofundar o conhe-

cimento em relação ao MPTPC em ambientes de nuvem com instâncias LXC e KVM, buscando outros resultados. O segundo ponto é testar todos os modos do protocolo *bonding*, tanto em ambiente nativo quanto em ambiente de nuvem com instâncias LXC e KVM, para então demonstrar as particularidades de cada modo.

REFERÊNCIAS

ABD, A. et al. LS-SCTP: a bandwidth aggregation technique for stream control transmission protocol. **Computer Communications**, [S.l.], v.27, n.10, p.1012–1024, 2004.

AL-FARES, M. et al. Hedera: dynamic flow scheduling for data center networks. In: NSDI. **Anais...** [S.l.: s.n.], 2010. v.10, p.19–19.

ANDRIOLI, L.; ROSA RIGHI, R. da; AUBIN, M. R. Analisando métodos e oportunidades em redes definidas por software (SDN) para otimizações de tráfego de dados. **Revista Brasileira de Computação Aplicada**, [S.l.], v.9, n.4, p.2–14, 2017.

AUST, S. et al. Evaluation of Linux bonding features. In: COMMUNICATION TECHNOLOGY, 2006. ICCT06. INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 2006. p.1–6.

AZEVEDO, E. et al. Nuvem pública versus privada: variações no desempenho de infraestrutura para elasticidade. **Anais do WCGA**, [S.l.], 2012.

BARRÉ, S. et al. Experimenting with multipath TCP. In: ACM SIGCOMM COMPUTER COMMUNICATION REVIEW. **Anais...** [S.l.: s.n.], 2010. v.40, n.4, p.443–444.

BARRÉ, S.; PAASCH, C.; BONAVENTURE, O. Multipath TCP: from theory to practice. In: INTERNATIONAL CONFERENCE ON RESEARCH IN NETWORKING. **Anais...** [S.l.: s.n.], 2011. p.444–457.

BERRANGé, D. P. **Manage virtual machines with virt-manager** <**https**://https://virt-manager.org/>. Último acesso Março de 2018.

BESERRA, D. et al. Performance analysis of LXC for HPC environments. In: COMPLEX, INTELLIGENT, AND SOFTWARE INTENSIVE SYSTEMS (CISIS), 2015 NINTH INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 2015. p.358–363.

BONDAN, L.; GOBBI, R. C.; FOCHI, V. M. **O Protocolo de Agregacão de Enlaces LACP**. Porto Alegre, RS, Brazil, 2012.

BRASSIL, J. An empirical study of bonded heterogeneous WAN access links. In: LOCAL AND METROPOLITAN AREA NETWORKS, 2005. LANMAN 2005. THE 14TH IEEE WORKSHOP ON. **Anais...** [S.l.: s.n.], 2005. p.6–pp.

BRIGHT, S. **UbuntuBonding** <**https**://help.ubuntu.com/community/ubuntubonding>. Último acesso Maio de 2018.

CHAPPELL, D. Introducing the windows azure platform. **David Chappell & Associates White Paper**, [S.l.], 2010.

CHIRAMMAL, H. D.; MUKHEDKAR, P.; VETTATHU, A. **Mastering KVM Virtualization**. [S.l.]: Packt Publishing Ltd, 2016.

CHO KENJIRO E CROVELLA, M. CoNEXT '11: proceedings of the seventh conference on emerging networking experiments and technologies. In: New York, NY, USA. **Anais...** ACM, 2011.

COSTA, G. H. d. **Métricas para avaliação de desempenho em redes QoS sobre IP**. 2008.

COUTINHO, E. et al. Elasticidade em computaçao na nuvem: uma abordagem sistemática. **XXXI Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuıdos (SBRC 2013)-Minicursos**, [S.l.], 2013.

DAVIS, T. et al. Linux ethernet bonding driver HOWTO. **available as download (bonding. txt) from the Linux Channel Bonding project Web site http://sourceforge. net/projects/bonding/(November 2007)**, [S.l.], 2011.

DEEK, L. et al. The impact of channel bonding on 802.11 n network management. In: SEVENTH CONFERENCE ON EMERGING NETWORKING EXPERIMENTS AND TECHNOLOGIES. **Proceedings...** [S.l.: s.n.], 2011. p.11.

EDIN, C. S. B. **Computação em nuvem e as diferenças tecnológicas entre o Amazon e o Azure.** [S.l.]: Universidade Tecnológica Federal do Paraná, 2011.

FELTER, W. et al. An updated performance comparison of virtual machines and linux containers. In: PERFORMANCE ANALYSIS OF SYSTEMS AND SOFTWARE (ISPASS), 2015 IEEE INTERNATIONAL SYMPOSIUM ON. **Anais...** [S.l.: s.n.], 2015. p.171–172.

FISHER, J. S. **Link aggregation on a LXC bridge Roadmap** <**https**://askubuntu.com/questions/492181/link-aggregation-on-a-lxc-bridge>. Last access May, 2018.

GIORGI, M. R. **Estudo comparativo sobre virtualização**. [S.l.: s.n.], 2013.

GONT, F. **ICMP attacks against TCP**. [S.l.: s.n.], 2010.

GOTO, Y. Kernel-based virtual machine technology. **Fujitsu Scientific and Technical Journal**, [S.l.], v.47, n.3, p.362–368, 2011.

HAT, R. **LibVirt Virtualization API** <**https**://libvirt.org/index.html/>. Último acesso Junho de 2018.

HINTERSTEINER, J. Melhores práticas para switches de rede gerenciados. In: **Anais...** [S.l.: s.n.], 2016.

HUANG, B.; BAUER, M.; KATCHABAW, M. Hpcbench-a Linux-based network benchmark for high performance networks. In: HIGH PERFORMANCE COMPUTING SYSTEMS AND APPLICATIONS, 2005. HPCS 2005. 19TH INTERNATIONAL SYMPOSIUM ON. **Anais...** [S.l.: s.n.], 2005. p.65–71.

HUANG, C. et al. Erasure Coding in Windows Azure Storage. In: USENIX ANNUAL TECHNICAL CONFERENCE. **Anais...** [S.l.: s.n.], 2012. p.15–26.

HUI, G. et al. A design and evaluation of Ethernet links bundling systems. In: ADVANCED INFORMATION NETWORKING AND APPLICATIONS, 2004. AINA 2004. 18TH INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 2004. v.2, p.313–316.

IMAIZUMI, H. et al. Power saving mechanism based on simple moving average for 802.3 ad link aggregation. In: GLOBECOM WORKSHOPS, 2009 IEEE. **Anais...** [S.l.: s.n.], 2009. p.1–6.

IPERF. **iPerf - The ultimate speed test tool for TCP, UDP and SCTP** <**https**://iperf.fr/>. Último acesso Fevereiro de 2018.

ISO/IEC17789. **Information technology — Cloud computing — Reference architecture**. [S.l.: s.n.], 2014.

JAIN, N. **Multichannel CSMA protocols for ad hoc networks**. 2001. Tese (Doutorado em Ciência da Computação) — University of Cincinnati.

KAUP, F. et al. Can Multipath TCP save energy? A measuring and modeling study of MPTCP energy consumption. In: LOCAL COMPUTER NETWORKS (LCN), 2015 IEEE 40TH CONFERENCE ON. **Anais...** [S.l.: s.n.], 2015. p.442–445.

KAVIS, M. **Architecting the Cloud**: design decisions for cloud computing service models (saas, paas, and iaas). [S.l.]: Wiley, 2014. (Wiley CIO).

KEDROWITSCH, A. et al. A First Look: using linux containers for deceptive honeypots. In: WORKSHOP ON AUTOMATED DECISION MAKING FOR ACTIVE CYBER DEFENSE, 2017. **Proceedings...** [S.l.: s.n.], 2017. p.15–22.

KHEIRKHAH, M.; WAKEMAN, I.; PARISIS, G. Multipath-TCP in ns-3. **arXiv preprint arXiv:1510.07721**, [S.l.], 2015.

KÖBEL, C.; BALUJA GARCÍA, W.; HABERMANN, J. Sistema de balance de carga para redes malladas inalámbricas multi-interfaces. **Ingeniería Electrónica, Automática y Comunicaciones**, [S.l.], v.33, n.3, p.85–98, 2012.

LAPUH, R.; HOMMA, T. **Routed split multilink trunking**. [S.l.]: Google Patents, 2008.

LIM, Y.-s. et al. Improving energy efficiency of mptcp for mobile devices. **arXiv preprint arXiv:1406.4463**, [S.l.], 2014.

LIM, Y.-s. et al. Design, implementation, and evaluation of energy-aware multi-path TCP. In: ACM CONFERENCE ON EMERGING NETWORKING EXPERIMENTS AND TECHNOLOGIES, 11. **Proceedings...** [S.l.: s.n.], 2015. p.30.

LIMA, D. C. S. F. **Soluções Alternativas de Escalonamento e Gerenciamento de Caminhos para o MPTCP**. 2017.

LOH, K. C. **Understanding Virtual Concatenation and Link Capacity Adjustment Scheme in SONET/SDH**. 2012. Tese (Doutorado em Ciência da Computação) — Thesis Naval Postgraduate School Monterey, California Issn.

LOVATO, A. **Metodologia da Pesquisa. Três de Maio**: setrem. [S.l.]: ISBN 978-85-99020-05-0, 2013.

LUO, Y. Network I/O virtualization for cloud computing. **IT professional**, [S.l.], v.12, n.5, p.36–41, 2010.

MATTEUSSI, K. J. et al. Avaliação de Desempenho Sobre Agregações de Links em Modo Round Robin. In: ESCOLA REGIONAL DE REDES DE COMPUTADORES (ERRC), 12. **Anais...** Sociedade Brasileira de Computação, 2014. p.1–6.

MATTOS, D. M. F. et al. **Um Mecanismo de Comutaç ao Multicaminhos de Pacotes Baseado em OpenFlow para Redes de Centros de Dados**. 2011.

MCILLECE, J.; POGGEMEYER, L. **Software Defined Networking (SDN)** <**https**://docs.microsoft.com/en-us/windows-server/networking/sdn/software-defined-networking>. Último acesso Fevereiro de 2018.

MEDEIROS, M. C. P. d. **Análise da implantação de computação em nuvem**: estudo de caso na alfa informatica. net–currais novos. 2015. B.S. thesis — Universidade Federal do Rio Grande do Norte.

MICROSOFT. **SDN - Software Defined Network** <**https**://www.microsoft.com/en-us/cloud-platform/software-defined-networking>. Último acesso Janeiro de 2018.

MING, L. et al. MPTCP incast in data center networks. **China Communications**, [S.l.], v.11, n.4, p.25–37, 2014.

NETPERF. **netperf - a network performance benchmark** <**https**://linux.die.net/man/1/netperf>. Último acesso Maio de 2018.

NGUYEN, S. C.; NGUYEN, T. M. T. Evaluation of multipath TCP load sharing with coupled congestion control option in heterogeneous networks. In: GLOBAL INFORMATION INFRASTRUCTURE SYMPOSIUM (GIIS), 2011. **Anais...** [S.l.: s.n.], 2011. p.1–5.

OKAMOTO, T. et al. RI2N/UDP: high bandwidth and fault-tolerant network for a pc-cluster based on multi-link ethernet. In: PARALLEL AND DISTRIBUTED PROCESSING SYMPOSIUM, 2007. IPDPS 2007. IEEE INTERNATIONAL. **Anais...** [S.l.: s.n.], 2007. p.1–8.

ONG, L. **An introduction to the stream control transmission protocol (SCTP)**. 2002.

OPENSTACK. **Basic architecture** <**https**://docs.openstack.org/glance/latest/contributor/architecture.html>. Last Acess in Fevereiro, 2018.

PITANGA, M. **Construindo supercomputadores com Linux**. [S.l.]: Brasport, 2004.

POPESKIC, V. **MPTCP – Multipath TCP** <**https**://howdoesinternetwork.com/2013/multipath-tcp>. Último acesso Fevereiro de 2018.

QEMU. **QEMU** <**https**://wiki.qemu.org/main_pagel//>. Último acesso Maio de 2018.

QIU, Y. **Evaluating and Improving LXC Container Migration between Cloudlets Using Multipath TCP**. 2016. Tese (Doutorado em Ciência da Computação) — Ph. D. Dissertation. Carleton University Ottawa.

RED HAT, I. **Openstack Identity (Keystine)** <**https**://access.redhat.com/documentation/en-us/red_hat_ enterprise_linux_openstack_platform/6/html/component_overview/section-identity.html>. Last Acess in March, 2018.

RISTA, C. et al. Improving the Network Performance of a Container-Based Cloud Environment for Hadoop Systems. In: HIGH PERFORMANCE COMPUTING & SIMULATION (HPCS), 2017 INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 2017. p.619–626.

ROVEDA, D. et al. Analisando a Camada de Gerenciamento das Ferramentas CloudStack e OpenStack para Nuvens Privadas. **13th Escola Regional de Redes de Computadores (ERRC), Passo Fundo, Brazil. Sociedade Brasileira de Computaçao**, [S.l.], 2015.

SALVO, H. Análise do impacto do isolamento em ambientes virtuais. In: **Anais...** [S.l.: s.n.], 2014.

SANTOS, C. F. **Ambiente de virtualização**: uma análise de desempenho. 2011.

SARABANDO, N. G. **Validation of"triple-play"services in the access node**. 2008. Dissertação (Mestrado em Ciência da Computação) — Universidade de Aveiro.

SARABANDO, N. G. Link Físico e Link Lógico. In: Três de Maio, RS, Brasil. **Anais...** [S.l.: s.n.], 2017.

SARABANDO, N. G. Arquitetura MPTCP. In: Três de Maio, RS, Brasil. **Anais...** [S.l.: s.n.], 2017.

SCHARF, A.-L. B. L.; FORD, C. **Multipath TCP (MPTCP) Application Interface Considerations** <**https**://www.rfc-editor.org/rfc/rfc6897.txt>. Último acesso Janeiro de 2018.

SHAMANI, M. J. et al. Signal aware multi-path TCP. In: WIRELESS ON-DEMAND NETWORK SYSTEMS AND SERVICES (WONS), 2016 12TH ANNUAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 2016. p.1–4.

SHI, J. et al. Experimental performance studies of SCTP in wireless access networks. In: COMMUNICATION TECHNOLOGY PROCEEDINGS, 2003. ICCT 2003. INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 2003. v.1, p.392–395.

SOTO, J. A. OpenNebula: implantação de uma nuvem privada e orquestração das máquinas virtuais no paradigma da computação em nuvem. **Monografia, Departamento de Engenharia de Teleinformática-Universidade Federal do Ceará-Fortaleza**, [S.l.], 2011.

SOUSA, F. R. et al. Gerenciamento de dados em nuvem: conceitos, sistemas e desafios. **Topicos em sistemas colaborativos, interativos, multimidia, web e bancos de dados, Sociedade Brasileira de Computação**, [S.l.], p.101–130, 2010.

SOUSA, F. R.; MOREIRA, L. o.; MACHADO, J. C. Computação em nuvem: conceitos, tecnologias, aplicações e desafios. **II Escola Regional de Computação Ceará, Maranhão e Piauí (ERCEMAPI)**, [S.l.], p.150–175, 2009.

TEKA, F.; LUNG, C.-H.; AJILA, S. Seamless live virtual machine migration with cloudlets and multipath TCP. In: COMPUTER SOFTWARE AND APPLICATIONS CONFERENCE (COMPSAC), 2015 IEEE 39TH ANNUAL. **Anais...** [S.l.: s.n.], 2015. v.2, p.607–616.

TRINDADE, L. V. P.; COSTA, L. H. M. Análise do Desempenho da Virtualização Leve para Ambientes com Edge Computing baseada em NFV. In: SIMPÓSIO BRASILEIRO DE REDES DE COMPUTADORES (SBRC). **Anais...** [S.l.: s.n.], 2018. v.36.

TUDORAN, R. et al. A performance evaluation of Azure and Nimbus clouds for scientific applications. In: INTERNATIONAL WORKSHOP ON CLOUD COMPUTING PLATFORMS, 2. **Proceedings...** [S.l.: s.n.], 2012. p.4.

UPERF. **Uperf - A network performance toolk** <**https**://http://uperf.org/>. Último acesso Março de 2018.

VAN DER POL, R. et al. Multipathing with MPTCP and OpenFlow. In: HIGH PERFORMANCE COMPUTING, NETWORKING, STORAGE AND ANALYSIS (SCC), 2012 SC COMPANION:. **Anais...** [S.l.: s.n.], 2012. p.1617–1624.

VOGEL, A. et al. Private IaaS clouds: a comparative analysis of opennebula, cloudstack and openstack. In: PARALLEL, DISTRIBUTED, AND NETWORK-

BASED PROCESSING (PDP), 2016 24TH EUROMICRO INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 2016. p.672–679.

VOGEL, A. et al. An intra-cloud networking performance evaluation on cloudstack environment. In: PARALLEL, DISTRIBUTED AND NETWORK-BASED PROCESSING (PDP), 2017 25TH EUROMICRO INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 2017. p.468–472.

WANG, G.; NG, T. E. The impact of virtualization on network performance of amazon ec2 data center. In: INFOCOM, 2010 PROCEEDINGS IEEE. **Anais...** [S.l.: s.n.], 2010. p.1–9.

WEN, X. et al. Comparison of open-source cloud management platforms: openstack and opennebula. In: FUZZY SYSTEMS AND KNOWLEDGE DISCOVERY (FSKD), 2012 9TH INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 2012. p.2457–2461.

WETHERALL, J.; TANENBAUM, A. Redes de Computadores. 5ª edição. **Rio de Janeiro: Editora Campus**, [S.l.], 2011.

WISCHIK, D. et al. Design, Implementation and Evaluation of Congestion Control for Multipath TCP. In: NSDI. **Anais...** [S.l.: s.n.], 2011. v.11, p.8–8.

ZHAI, E.; CUMMINGS, G. D.; DONG, Y. Live migration with pass-through device for Linux VM. In: OLS'08: THE 2008 OTTAWA LINUX SYMPOSIUM. **Anais...** [S.l.: s.n.], 2008. p.261–268.

ZHOU, F. et al. The Performance Impact of Buffer Sizes for Multi-path TCP in Internet Setups. In: ADVANCED INFORMATION NETWORKING AND APPLICATIONS (AINA), 2017 IEEE 31ST INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 2017. p.9–16.