FuzSy: Um Escalonador baseado em Qualidade de Serviço e Lógica Fuzzy

Fabrício R. de Souza[1], **Fátima Duarte-Figueiredo**[1]

[1]Departamento de informática - Pontifícia Universidade Católica de Minas Gerais (PUC Minas)

fabricio127@gmail.com, fatimafig@pucminas.br

Abstract. *In this paper, FuzSy, or Fuzzy Scheduler, is presented. It is a cellular network packet scheduler. It is based on quality of service and fuzzy logic. Classic solutions utilize static packet scheduling, following a prearranged priority order. Static scheduling can be unfair, or waste network resources, trying to allocate resources. FuzSy utilizes metadata from the network and, from choices made by the fuzzy logic mechanism, which seeks to prioritize classes that have a higher risk of not reaching it's QoS requirements, sets a dynamic packet priority. This priority is said to be dynamic because, is one class is in a QoS position that is not favorable, the scheduler set this class' priority to be higher, regardless of any previous set priority. Fourth generation, 4G, network simulations were conducted. The results show that FuzSy was able to keep all QoS within 3GPP stipulated values and was able to achieve higher fairness, not benefiting any class at other's cost. One of FuzSy's advantages, in relation to the other schedulers, cited in this paper, is that parameters can be added to or deleted from the fuzzy mechanism. With that, on specific situations such as places hosting large scale events the scheduler can be altered and utilize different parameters and rules to better serve the users.*

Resumo. *Neste trabalho, é apresentado o FuzSy, ou Fuzzy Scheduler, um escalonador de pacotes para redes de celular. Ele é baseado em qualidade de serviço e lógica fuzzy. Soluções clássicas utilizam escalonamento estático de pacotes, seguindo prioridades previamente negociadas. O escalonamento estático pode ser injusto ou, até mesmo, pode desperdiçar recursos da rede. O FuzSy utiliza metadados da rede e, a partir da escolha feita pelo mecanismo de lógica fuzzy, que visa dar maior prioridade à classes que correm mais risco de não atingirem seus requisitos de QoS, atribui uma prioridade dinâmica aos pacotes. Essa prioridade é dinâmica porque, se uma classe se encontra em condições de QoS desfavoráveis, o escalonador passa a dar maior prioridade a ela, independentemente de qualquer atribuição feita anteriormente. Simulações de uma rede de quarta geração, 4G, foram realizadas. Os resultados mostraram que o FuzSy conseguiu manter todas as métricas de QoS dentro dos limites estipulados pelo 3GPP e conseguiu ser mais justo, não beneficiando nenhuma classe em detrimento de outras. Uma vantagem do FuzSy, em relação aos demais escalonadores, citados no trabalho, é que podem ser adicionados ou excluídos novos parâmetros e regras no mecanismo de lógica fuzzy. Com isso, situações específicas como locais com eventos de larga escala podem precisar de escalonadores com regras adaptáveis para atender melhor aos usuários.*

1. Introdução

Nas últimas décadas, a quantidade de usuários das redes de celular vem crescendo vertiginosamente e a tendência é que esse ritmo de crescimento continue. De acordo com dados disponibilizados pela Anatel (Agência Nacional de Telecomunicações) [IBGE 2016], em 2016, a taxa de dispositivos móveis por habitante no Brasil era de 1,26, o que demonstra a necessidade de acesso de qualidade a essas redes.

As redes LTE, implantadas e em uso em todo o mundo, também chamadas de redes 4G (de quarta geração), são redes de celulares que permitem acesso a diversos tipos de serviços, como, por exemplo, transmissão de voz, navegação na web, transmissão de conteúdo multimídia em tempo real, entre outros. No mundo atual, onde a quantidade de usuários simultâneos dessas redes é muito grande, é necessário que a transmissão dos pacotes seja controlada dinamicamente, para que seja possível garantir qualidade de serviço nas redes.

Numa rede com serviços diferentes, é necessário que cada um destes serviços seja tratado de forma a garantir que seus requisitos de QoS específicos sejam atendidos. Certos tipos de serviço são mais sensíveis ao atraso dos pacotes, enquanto outros dependem mais da vazão disponível, o que significa que tratar os serviços de forma igual não é interessante, visto que eles têm requisitos diferentes. Uma das maneiras de medir se uma classe de serviço foi, ou não, atendida é através da métrica de equidade. A equidade mede o quão bem um requisito de QoS foi atendido. Através do cálculo da equidade, verifica-se se as classes mais prioritárias da rede estão sendo bem atendidas, sem prejudicar as menos prioritárias, refletindo critérios justos de priorização de acesso e escalonamento na rede.

Para garantir a qualidade de serviço e a equidade entre as classes, existem mecanismos para ajustar o nível de envios de pacotes da rede. Um exemplo desses mecanismos é um escalonador de pacotes, que é responsável por decidir como e quando os pacotes da rede serão enviados. Neste trabalho, é apresentado o FuzSy (*Fuzzy Scheduler*) um escalonador de pacotes que se baseia em métricas de qualidade de serviço e um mecanismo de lógica *fuzzy* para escalonar os pacotes. O FuzSy foi proposto em uma dissertação de mestrado [de Souza 2016]. Normalmente, um escalonador utiliza uma fórmula matemática para o cálculo da prioridade de cada um dos pacotes. Diferentemente, o FuzSy utiliza um mecanismo inteligente baseado no estado atual da rede para decidir qual pacote é o mais prioritário em um determinado momento. Prioridades estáticas são prioridades que são definidas uma vez e nunca modificadas durante o envio dos pacotes. O FuzSy utiliza o conceito de prioridades dinâmicas, que se alteram, à medida que é necessário, durante o envio dos pacotes.

O FuzSy atingiu seu objetivo de prover maior equidade para as classes, conforme mostram resultados de simulações. Para todas as classes, o FuzSy foi o escalonador que apresentou maior equidade e na maioria dos casos foi o escalonador que atingiu melhores resultados para os parâmetros mais sensíveis de cada classe, quando comparado a outras propostas. O restante deste trabalho está dividido da seguinte maneira: sessão 2 elabora sobre os escalonadores de pacote existentes na literatura; a sessão 3 descreve o escalonador FuzSy; a sessão 4 apresenta os resultados obtidos através de simulações; a sessão 5 contém as conclusões do trabalho.

2. Escalonadores de pacote

Em qualquer rede de telefonia celular, na estação rádio base, existe uma fila com os pacotes que devem ser enviados para os usuários. Essa fila consiste em pacotes de diversas classes de serviço. É papel do escalonador de pacotes estabelecer a prioridade de cada pacote e reorganizar a fila de acordo essa prioridade. É necessário que o processo de escalonamento seja feito para que a qualidade de serviço da rede seja mantida, ou seja, para que uma classe não fique sem atendimento adequado. A maioria dos escalonadores de pacotes existentes na literatura utilizam fórmulas matemáticas para o cálculo das prioridades. Nesta seção são citados alguns dos escalonadores existentes na literatura.

Em [Jalali et al. 2000] o escalonador *Proportional Fair* é definido. Ele tenta maximizar a vazão da rede, não se preocupando com qualquer outro parâmetro, e utiliza a Equação 1 para calcular sua métrica.

$$P = \frac{T^{\alpha}}{R^{\beta}} \tag{1}$$

Na equação, T é a taxa de transferência disponível na antena que enviou o pacote naquele momento e R é a média da taxa de transferência daquela antena até o momento. Os valores de α e β são utilizados para regular a execução do algoritmo. Se o valor de α for escolhido como 0 e o de β como 1, o algoritmo executará como *roundrobin*, enviando todos os pacotes na ordem que eles chegarem, desconsiderando a qualidade dos canais. Se os valores forem invertidos, α como 1 e β como 0, o algoritmo sempre enviará os pacotes com a melhor qualidade de canal, ou seja, a vazão da rede será maximizada.

O escalonador FLS (*Frame Level Scheduler*) [Piro et al. 2011b] concentra-se no envio de pacotes de tempo real, calculando a quantidade necessária de recursos que deve ser alocada para que todos os pacotes de tempo real sejam enviados dentro dos seus requisitos de atraso, independentemente da quantidade de pacotes de outras classes que existam na fila de envio. Os escalonadores EXP-Rule (*Exponential Rule*) [Shakkottai and Stolyar 2002], LOG-Rule (*Loharithm Rule*) [Sadiq et al. 2011], EXP (*Exponential Proportional Fair*) [Rhee et al. 2004] e MLWDF (Modified Largest Weighted Delay First) [Andrews et al. 2001] tem o intuito de melhorar o envio dos pacotes de tempo real, adicionando um peso à métrica calculada pelo *Proportional fair*, o peso calculado por cada um está disponível na Tabela 1.

Tabela 1. Pesos dos escalonadores

Escalonador	Peso
EXP-Rule	$\sigma = exp(\frac{\sqrt[6]{\text{Atraso desejado}}^{Hol}}{1+\sqrt{\text{Média dos Hol}}})$
LOG-Rule	$\sigma = log(1.1 + (\frac{5Hol}{\text{Atraso desejado}}))$
EXP	$\sigma = exp(\frac{\gamma - Z}{1 - \sqrt{Z}}) \quad Z = \frac{\gamma HOL - \frac{1}{N_{RT}}\sum_{t=1}^{N_{RT}} \gamma HOL}{N_{RT}}$
MLWDF	$\sigma = -\frac{\log \delta}{\tau} T_{Max}$

A função exp(x) calcula o valor de e^x, a função log(x) retorna o logaritmo natural de x, HOL é o maior valor de atraso de pacotes de tempo real, Atraso desejado é o atraso máximo permitido para aquela classe, N_{RT} representa a quantidade de pacotes de tempo

real a serem enviados, T_{Max} representa o T máximo atingido por aquela antena, δ representa o tempo que o pacote passou na fila até o momento e τ representa o maior tempo que um pacote passou na fila.

Uma das maneiras apresentadas para a definição de parâmetros para o escalonamento é a utilização de *pricing models*. *Pricing model* é um modelo que segue diversas regras e atribuem um preço a um pacote. Esse preço pode ser entendido como o custo para a rede enviar o pacote ou o ganho que a rede tem se ele for enviado naquele instante. Diversos *pricing models* utilizam diferentes estratégias para criar o preço de um pacote. Em [Wallenius and Hamalainen 2002] os autores utilizam uma equação que se baseia em variáveis que descrevem o estado da rede como, por exemplo, vazão, atraso, *jitter* etc para elaborar o preço de cada um dos pacotes. A cada classe da rede, é atribuída uma prioridade. A ordem desta prioridade é: conversacional, *streaming*, interativas e de *background*. Além disso, também é criado um valor fixo para o custo de transmitir 1 bit daquele tipo de classe. O valor do preço dos bits é multiplicado pela constante que representa a prioridade da classe e esse é o preço para o envio daquele pacote. No entanto, o preço pode ser ajustado por um fator chamado de ”linearidade”pelos autores. Esse fator altera o preço do pacote se os recursos da rede estiverem escassos o preço do pacote sobe e ele pode não ser enviado no momento, porque o custo é alto para a rede.

Um *pricing model* tenta criar um equilíbrio entre a quantidade de recursos utilizados na rede e a qualidade das chamadas feitas. O trabalho apresentado em [Lai and Jiang 2014] utiliza um algoritmo genético que tenta encontrar a melhor solução de compromisso para o problema. Os autores utilizam duas equações para descrever o problema. Uma que descreve o lucro, ou seja, a quantidade de recursos que foi utilizada, e outra que descreve a satisfação dos usuários, ou seja, a qualidade de serviço da rede. O resultado do algoritmo genético é uma equação que descreve como cada classe de pacotes deve ter seu preço atribuído. Os resultados apresentados no artigo demonstram que as variáveis utilizadas nas equações são inversamente proporcionais, por isso é necessário encontrar um ponto de equilíbrio entre as duas. O algoritmo genético utilizado pelos autores consegue, a partir de um preço máximo para cada classe de serviço e uma satisfação mínima que deve ser atingida, criar um modelo que tenta balancear a satisfação e a utilização.

Em [Tarchi et al. 2006], os autores utilizam duas filas de prioridade para o envio dos pacotes. É definida uma prioridade para cada classe de serviço da rede e limites mínimos que ditam a porcentagem mínima de pacotes daquela classe devem ser enviados. A primeira fila de prioridades é a fila das classes de serviço. Enquanto o limiar de uma classe não for quebrado, a mais prioritária, com o limiar mais baixo é escolhida para ser enviada. Quando um limiar é quebrado aquela classe é escolhida para ser enviada enquanto o limiar não voltar para um nível aceitável. Após a escolha da classe, os autores escalonam os vários pacotes daquela classe. Eles utilizam técnicas clássicas como *round-robin* e *proportional fairness* para o envio dos pacotes.

Em [Ball et al. 2005], os autores utilizam uma fila simples de prioridade para escalonar os pacotes. A diferença apresentada no artigo é que os autores removem da fila pacotes que apresentam sinal de transmissão ruim. O autores fazem isso com a ideia de que pacotes que apresentam um nível ruim da qualidade de sinal tem grande chance de se perderem durante. Para evitar o envio de um pacote que será provavelmente perdido, os

autores o removem da lista por um tempo pré determinado e depois o colocam novamente. Esse processo é repetido algumas vezes enquanto o sinal for considerado ruim. Após um certo número de tentativas, o pacote é enviado de qualquer maneira.

Outra técnica utilizada é a apresentada em [Pizzi et al. 2009], na qual os autores estudam o que acontece se apenas um parâmetro da rede for utilizado para o escalonamento. Os pacotes são colocados em uma fila de prioridade e aqueles que possuem o maior *jitter* são enviados primeiro. O artigo demonstra que classes sensíveis ao *jitter* como a de *streaming* são extremamente beneficiadas por essa técnica. Contudo as classes que dependem de outros parâmetros da rede acabam sendo prejudicadas. Os autores em [Lai and Tang 2013] utilizam um modelo de de previsão de envio de pacotes para fazer o escalonamento. São utilizadas diversas variáveis de QoS nos cálculos do modelo como, por exemplo, atraso, *jitter* etc. O intuito do modelo é calcular se dado pacote que está na fila de envio chegará ou não a tempo, com atraso aceitável, ao seu destino. Uma vez que a previsão dos pacotes é feita, a fila de envio é reorganizada colocando os que atendem à condição na frente.

3. FuzSy: Escalonador de Pacotes baseado em QoS e Lógica *Fuzzy*

A ideia principal de um escalonador é despachar pacotes que chegam em elementos centralizadores de uma rede, tais como roteadores ou estações rádio base. Os escalonadores normalmente utilizam filas de pacotes e atribuem prioridades aos mesmos. Do mecanismo mais simples, FIFO, aos descritos anteriormente, a maioria segue regras estáticas de escolha de pacotes prioritários. Como em uma rede de celulares os picos de utilização dependem de horário, localização e dia da semana, os critérios estáticos podem significar uma distribuição injusta de recursos. Para tornar a distribuição mais justa, a prioridade de escalonamento deve ser alterada, dinamicamente, de acordo com a demanda do momento.

O FuzSy, ou *Fuzzy Scheduler*, se baseia no estado atual da rede e utiliza um mecanismo de lógica *fuzzy* para decidir qual classe de serviço é mais prioritária em dado instante. Para tomar essa decisão, o FuzSy utiliza metadados da rede, dados coletados durante a transmissão dos pacotes. Os metadados coletados são o atraso médio, que representa o tempo médio gasto para um pacote sair do dispositivo origem e chegar ao dispositivo destino, o *jitter* médio, que representa a diferença do atraso médio entre dois pacotes consecutivos da mesma aplicação, a vazão média, que representa a quantidade média de dados transmitidos em dado tempo, e a quantidade de pacotes a serem enviados. Cada uma das métricas citadas são coletadas separadamente para cada classe de serviço. As métricas coletadas são separadas por classe de serviço, porque cada uma destas classes apresenta um requisito diferente sobre essas métricas, ou seja, uma métrica tem maior impacto na qualidade daquele tipo de serviço do que em outro. Após a coleta dos metadados é necessário definir o que eles significam. Cada um dos metadados coletados possui três níveis, baixo, médio e alto. As Tabelas 2, 3, 4 e 5 possuem os valores utilizados para definição dos níveis.

Uma vez que todos os valores de entrada do sistema têm seus níveis definidos, são testadas as regras de inferência para decidir quais ações devem ser tomadas. Classicamente, num sistema de lógica *fuzzy*, cria-se todas as regras de inferência possíveis para o sistema, o que significaria criar 3^{16} regras neste sistema, o que é impraticável do ponto de vista de processamento, para isso foram criadas essas regras utilizando conjunções e

disjunções, para testar englobar todas as possíveis regras num conjunto menor. Na Tabela 6 estão algumas das regras utilizadas no sistema, por questões de espaço serão apresentadas apenas 5 das 27 regras utilizadas, mas todas seguem o mesmo padrão.

Tabela 2. Níveis de pertinência - Atraso médio

	Atraso médio (ms)		
	Baixo	Médio	Alto
Conversacional	0 - 75	25 - 100	50 - 150
Streaming	0 - 150	50 - 200	100 - 300
Interativa	1 - 150	50 - 200	101 - 300
Background	2 - 150	50 - 200	102 - 300

Tabela 3. Níveis de pertinência - *Jitter* médio

	Jitter Médio (ms)		
	Baixo	Médio	Alto
Conversacional	0 - 40	20 - 60	40 - 80
Streaming	0 - 20	10 - 30	20 - 40
Interativa	0 - 40	20 - 60	40 - 80
Background	0 - 40	20 - 60	40 - 80

Tabela 4. Níveis de pertinência - Vazão média

	Vazão Média (kbps)		
	Baixo	Médio	Alto
Conversacional	0 - 30	30 - 50	50 - 64
Streaming	0 - 350	150 - 665	500 - 1000
Interativa	0 - 150	100 - 250	200 - 350
Background	0 - 1500	1000 - 3000	2000 - 4500

Tabela 5. Níveis de pertinência - Quantidade de pacotes na fila

	Quantidade de pacotes (unidade)		
	Baixo	Médio	Alto
Conversacional	0 - 4	2 - 6	4 - 8
Streaming	0 - 4	2 - 6	4 - 8
Interativa	0 - 4	2 - 6	4 - 8
Background	0 - 4	2 - 6	4 - 8

Após o mecanismo de lógica *fuzzy* verificar todas as regras o processo de deffuzyficação é iniciado. Neste trabalho, foi utilizado o mecanismo chamado "maior de todos". Se mais de uma regra, para a mesma variável de saída, for atendida, a regra com o maior valor de pertinência é dada como válida e o valor é atribuído. Por exemplo, se uma regra disser que a variável deveria ser alta e outra regra disser que ela deveria ser muito alta o valor atribuído à variável é de muito alta. Durante as simulações notou-se que em ocasiões em que todas as variáveis de saída possuíam valores muito altos o escalonador não conseguia priorizar de forma correta o envio dos pacotes. Para isso, foi implementado um mecanismo estático de prioridades para ser executado apenas nesse caso. O mecanismo assumia que as classe seguiam a seguinte prioridade: conversacional,

streaming, interativa e *background* [3GPP 2013]. Uma vez que os níveis de prioridade das classes abaixassem, o mecanismo *fuzzy* entrava em ação novamente.

Tabela 6. Regras de inferência

	Regra
1	if (JitterStreaming is Alto and FilasStreaming is Alto) and (VazaoStreaming is any or DelayStreaming is any) then PrioridadeStreaming is MuitoAlto
2	if (JitterStreaming is Medio and FilasStreaming is Alto) and (VazaoStreaming is Baixo or DelayStreaming is Alto) then PrioridadeStreaming is Alto
3	if (JitterStreaming is Medio and FilasStreaming is Medio) and (VazaoStreaming is Baixo or DelayStreaming is Alto) then PrioridadeStreaming is Alto
4	if (JitterStreaming is Baixo and FilasStreaming is Baixo) and (VazaoStreaming is Baixo or DelayStreaming is Alto) then PrioridadeStreaming is Medio
5	if (JitterStreaming is Baixo and FilasStreaming is Baixo) and (VazaoStreaming is Alto or DelayStreaming is Baixo) then PrioridadeStreaming is Baixo

O processo descrito é executado para todos os pacotes que estão na fila de envio e, depois, a fila é reorganizada seguindo as prioridades definidas, uma prioridade maior é mais prioritária que uma menor. Sendo assim os pacotes que possuem nível de pertinência maior são os primeiros a saírem da fila.

4. Simulações e Resultados

Para avaliação do FuzSy, foram executadas simulações, utilizando o simulador LTE-Sim [Piro et al. 2011a]. O cenário de simulação foi composto por sete células de formato hexagonal. A quantidade inicial de usuários por célula foi 5 e essa quantidade foi sendo gradativamente aumentada até 25 usuários. Os usuários sempre possuíam acesso à rede e se movimentavam de acordo com um modelo de mobilidade aleatório a 3 km/h.

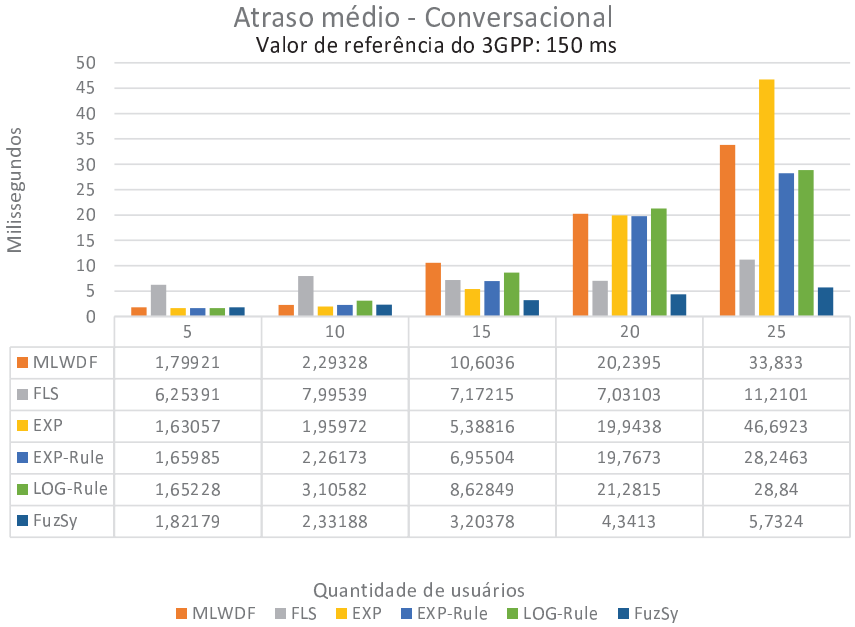
Tabela 7. Distribuição das aplicações

Classe	Distribuição
Conversacional	42%
Streaming	16%
Interativa	18.5%
Background	23.5%

Durante as simulações, foram criados quatro tipos de aplicações, uma para cada classe de serviço existente. Para representar a classe conversacional, foi simulada a transmissão de voz, pela rede, entre dois usuários. Para a classe *streaming*, foi utilizado a transmissão sob demanda em tempo real de um vídeo em 720p a partir de um servidor. Para a classe interativa, foi simulado o *download* de uma página da *web* com tamanho variando aleatoriamente entre 2 e 8 mb e para a classe *background*, foi simulado

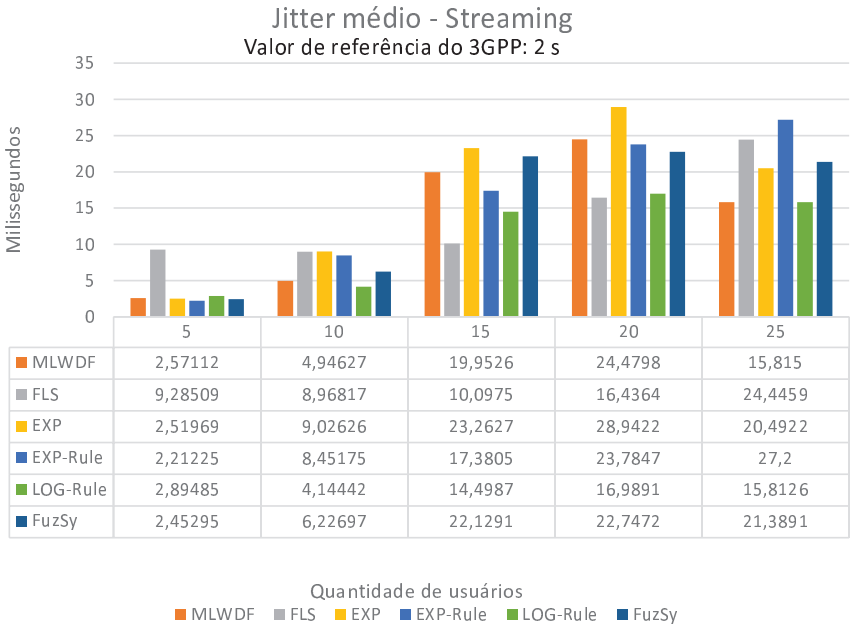
o download de um arquivo de um servidor com tamanho variando entre 10 e 50 mb. A distribuição das aplicações foi a mesma seguida em [Tostes Ribeiro et al. 2013], apresentada na Tabela 7, que representa uma distribuição mais próxima da realidade. Além dessa distribuição, também foi simulado um cenário com uma distribuição de 25% para cada tipo de aplicação e os resultados foram extremamente similares.

As métricas avaliadas foram o atraso médio dos pacotes, o *jitter* médio dos pacotes, a vazão média e a equidade média. Em todos os gráficos apresentados, os dados do escalonador *proportional fair* foram omitidos para melhor visualização, uma vez que ele sempre apresentava valores piores do que todos os outros escalonares avaliados, porque utiliza uma estratégia simples para fazer o escalonamento.



	5	10	15	20	25
MLWDF	1,79921	2,29328	10,6036	20,2395	33,833
FLS	6,25391	7,99539	7,17215	7,03103	11,2101
EXP	1,63057	1,95972	5,38816	19,9438	46,6923
EXP-Rule	1,65985	2,26173	6,95504	19,7673	28,2463
LOG-Rule	1,65228	3,10582	8,62849	21,2815	28,84
FuzSy	1,82179	2,33188	3,20378	4,3413	5,7324

Figura 1. Atraso médio - Conversacional



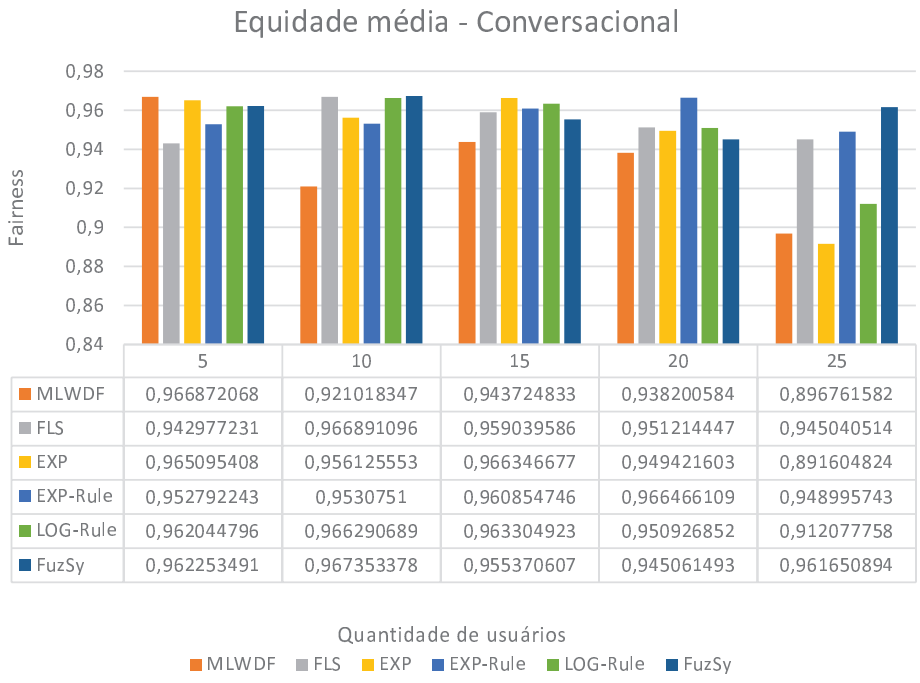
	5	10	15	20	25
MLWDF	2,57112	4,94627	19,9526	24,4798	15,815
FLS	9,28509	8,96817	10,0975	16,4364	24,4459
EXP	2,51969	9,02626	23,2627	28,9422	20,4922
EXP-Rule	2,21225	8,45175	17,3805	23,7847	27,2
LOG-Rule	2,89485	4,14442	14,4987	16,9891	15,8126
FuzSy	2,45295	6,22697	22,1291	22,7472	21,3891

Figura 2. *Jitter* médio - *Streaming*

No gráfico da Figura 1, pode-se observar o atraso médio para a classe conversacional. Pode-se perceber que o FuzSy apresentou resultados significativamente melhores que todos os outros, tendo valores máximos de aproximadamente 5 milissegundos para o cenário de 25 usuários que era o que apresentava maior utilização da rede.

O gráfico da Figura 2 mostra os valores de *jitter* para a classe *streaming*. O LOG-Rule foi o que apresentou melhores resultados neste quesito. O intuito do FuzSy não é conseguir os melhores valores em todos os parâmetros avaliados, deseja-se que ele consiga distribuir os recursos de forma mais justa, ou seja, sem prejudicar outras classes para priorizar uma. Devido a isso, ele não apresenta, necessariamente, o melhor ganho de desempenho, mas isso acontece para que seja possível garantir uma melhor distribuição dos recursos, o que o torna o mais justo de maneira geral.

No gráfico da Figura 3 estão os valores de equidade para a classe conversacional. Nele, é possível observar que para os cenários com até 20 usuários por célula todos os escalonadores conseguiram manter uma equidade acima de 90%. No entanto, no cenário com 25 usuários é possível perceber que o FuzSy apresentou os melhores resultados, seguido do EXP-Rule e, em seguida, pelo FLS. Isso demonstra que o FuzSy conseguiu lidar de uma maneira mais eficiente com a distribuição dos recursos para esta classe.



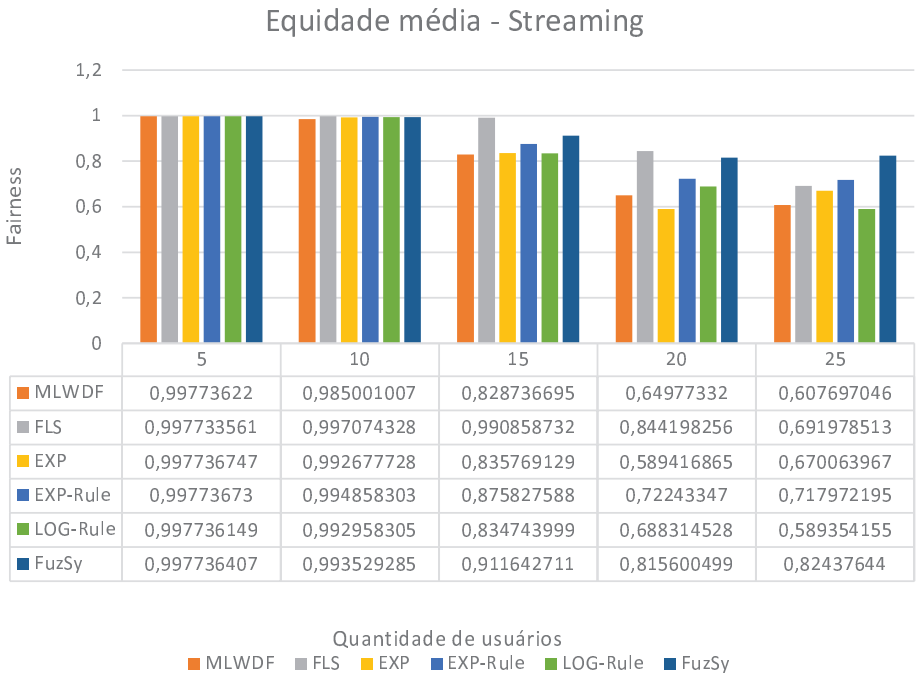
	5	10	15	20	25
MLWDF	0,966872068	0,921018347	0,943724833	0,938200584	0,896761582
FLS	0,942977231	0,966891096	0,959039586	0,951214447	0,945040514
EXP	0,965095408	0,956125553	0,966346677	0,949421603	0,891604824
EXP-Rule	0,952792243	0,9530751	0,960854746	0,966466109	0,948995743
LOG-Rule	0,962044796	0,966290689	0,963304923	0,950926852	0,912077758
FuzSy	0,962253491	0,967353378	0,955370607	0,945061493	0,961650894

Figura 3. Equidade média - Conversacional

Os valores de equidade média para a classe *streaming* estão no gráfico da Figura 4. É possível observar que, nos cenários até 10 usuários todos os escalonares ficaram muito próximos a 100% de equidade. A partir do cenário com 15 usuários esses valores começam a diminuir, sendo que para o cenário com 25 usuários, o mais denso, o FuzSy conseguiu uma equidade maior do que todos os outros escalonadores.

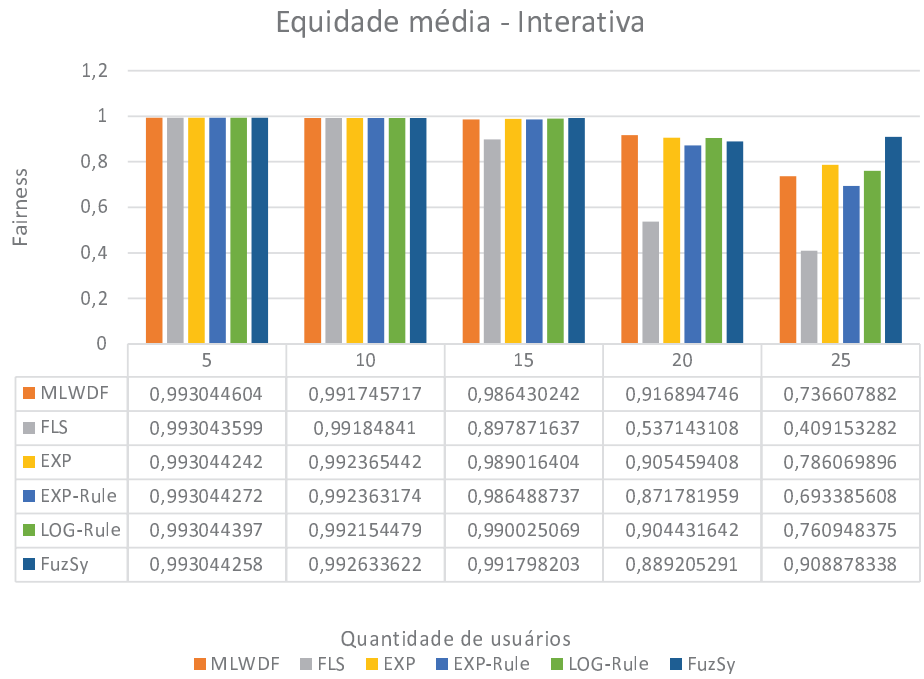
É possível observar os valores de equidade média para a classe interativa no gráfico da Figura 5. Os valores de equidade para o escalonador FLS caem significativamente para os cenários com um número maior de usuários por células. Isso acontece

devido à prioridade que este escalonador dá às classes conversacional e *streaming*. Com isso a classe interativa acaba ficando prejudicada. Diferentemente o FuzSy apresentou os maiores valores de equidade para os cenários com mais usuários por células, não prejudicando nenhuma classe, tentando manter parâmetros de QoS relevantes em níveis adequados para todas as classes.



	5	10	15	20	25
MLWDF	0,99773622	0,985001007	0,828736695	0,64977332	0,607697046
FLS	0,997733561	0,997074328	0,990858732	0,844198256	0,691978513
EXP	0,997736747	0,992677728	0,835769129	0,589416865	0,670063967
EXP-Rule	0,99773673	0,994858303	0,875827588	0,72243347	0,717972195
LOG-Rule	0,997736149	0,992958305	0,834743999	0,688314528	0,589354155
FuzSy	0,997736407	0,993529285	0,911642711	0,815600499	0,82437644

Figura 4. Equidade média - *Streaming*

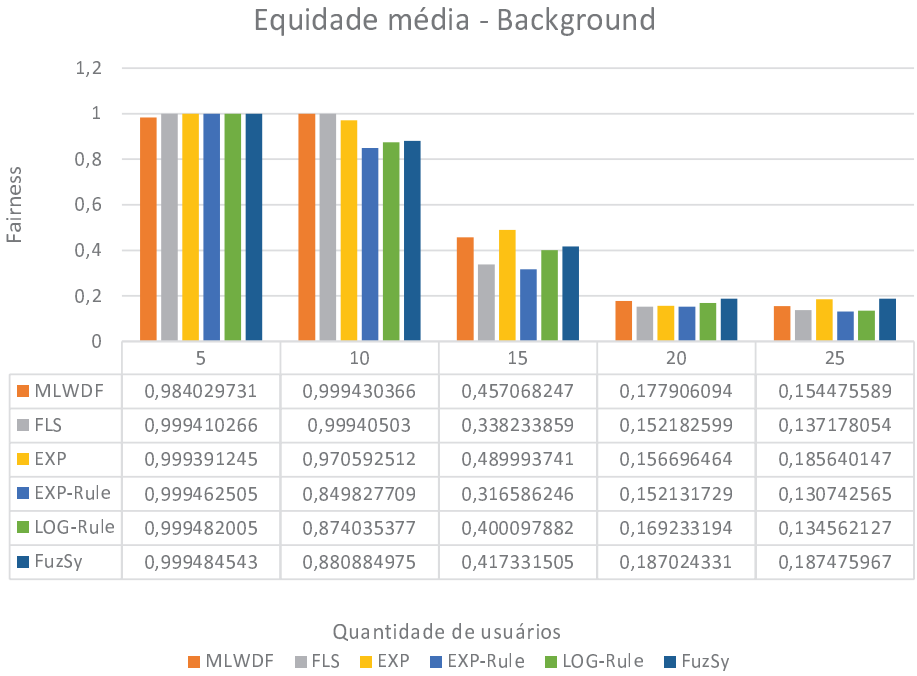


	5	10	15	20	25
MLWDF	0,993044604	0,991745717	0,986430242	0,916894746	0,736607882
FLS	0,993043599	0,99184841	0,897871637	0,537143108	0,409153282
EXP	0,993044242	0,992365442	0,989016404	0,905459408	0,786069896
EXP-Rule	0,993044272	0,992363174	0,986488737	0,871781959	0,693385608
LOG-Rule	0,993044397	0,992154479	0,990025069	0,904431642	0,760948375
FuzSy	0,993044258	0,992633622	0,991798203	0,889205291	0,908878338

Figura 5. Equidade média - Interativa

O gráfico da Figura 6 apresenta os valores de equidade média para a classe *background*. É possível observar que, à medida que a quantidade de usuários aumenta, a

equidade em todos os escalonadores diminui vertiginosamente. Isso acontece devido à disputa por recursos ser maior nestes cenários, com uma quantidade maior de usuários utilizando a rede os recursos que seriam alocados para esta classe são alocados para as aplicações das outras classes.



	5	10	15	20	25
MLWDF	0,984029731	0,999430366	0,457068247	0,177906094	0,154475589
FLS	0,999410266	0,99940503	0,338233859	0,152182599	0,137178054
EXP	0,999391245	0,970592512	0,489993741	0,156696464	0,185640147
EXP-Rule	0,999462505	0,849827709	0,316586246	0,152131729	0,130742565
LOG-Rule	0,999482005	0,874035377	0,400097882	0,169233194	0,134562127
FuzSy	0,999484543	0,880884975	0,417331505	0,187024331	0,187475967

Figura 6. Equidade média - *Background*

Na Tabela 8 estão as posições alcançadas pelos escalonadores quando comparados em relação aos parâmetros mais sensíveis de cada classe e também em relação à equidade, as letras após o nome do parâmetro representam a inicial de cada uma das classes. É possível perceber que, para a maioria dos parâmetros comparados, o FuzSy foi o escalonador com o melhor desempenho. Isso acontece graças a habilidade do FuzSy de se adaptar melhor às diversas situações que podem ocorrer. Ele não foi o escalonador com melhores resultados de *jitter* para a classe *streaming*, mas conseguiu garantir uma equidade maior do que todos os outros escalonadores.

Tabela 8. Ranking dos escalonadores para cenário com 25 usuários por célula

	1º	2º	3º	4º	5º	6º
Vazão I	FuzSy	EXP	LOG-Rule	MLWDF	EXP-Rule	FLS
Atraso C	FuzSy	FLS	EXP-Rule	LOG-Rule	MLWDF	EXP
Jitter S	LOG-Rule	MLWDF	EXP	FuzSy	FLS	EXP-Rule
Equidade C	FuzSy	EXP-Rule	FLS	LOG-Rule	MLWDF	EXP
Equidade S	FuzSy	EXP-Rule	FLS	EXP	MLWDF	LOG-Rule
Equidade I	FuzSy	EXP	LOG-Rule	MLWDF	EXP-Rule	FLS
Equidade B	FuzSy	EXP	MLWDF	FLS	LOG-Rule	EXP-Rule

5. Conclusão

O FuzSy se baseia no estado atual da rede e busca garantir a equidade entre os requisitos de QoS das classes de serviço da rede, através de controle dinâmico implementado por

meio de um mecanismo de lógica *fuzzy*. Utilizando metadados, colhidos durante o envio dos pacotes, o mecanismo de lógica *fuzzy* é executado para definir a prioridade dos pacotes que devem ser enviados. Tendo como entrada, para este mecanismo, parâmetros definidos através de simulações exaustivas e recomendações do órgão responsável pela padronização de redes 4G, ele consegue saber quais classes de serviço estão em maior risco de não atingirem seus limites mínimos de QoS. As regras de inferência criadas tentam fazer com que os parâmetros mais importantes para cada classe de serviço sejam levados em consideração, fazendo com que estes parâmetros sejam utilizados para melhorar a equidade entre as classes. A avaliação do escalonador *fuzzy* foi feita através de simulações realizadas no LTE-SIM. Ao ser comparado com seis outros métodos da literatura, o FuzSy apresentou resultados para todas as métricas dentro dos limites sugeridos pelo 3GPP e apresentou também melhorias na equidade das classes.

O diferencial do FuzSy, em relação aos outros escalonadores simulados, é a utilização de uma prioridade dinâmica para as classes. Os escalonadores avaliados definem quais classes de tempo real têm maior prioridade sobre as outras classes, o que pode causar má distribuição dos recursos. Foi demonstrado, através dos resultados de simulação, que a utilização de prioridades estáticas para as classes *streaming* e conversacional levaram o escalonador FLS a prejudicar a classe interativa, o que não acontece com o FuzSy. Sendo dinâmico, ele permite a utilização mais justa da rede, sem prejudicar o desempenho das mais prioritárias. Isso significa que há uma utilização mais efetiva e justa dos recursos da rede.

A utilização de um mecanismo de lógica *fuzzy* permite que os limites estabelecidos para cada classe sejam facilmente modificados e as regras de inferência também. O escalonador pode ser adaptado a qualquer situação específica que de uma operadora, como, por exemplo, eventos de larga escala, onde predomina um tipo ou dois tipos de aplicações ou classes de serviço. O FuzSy pode ser considerado mais justo do que os outros escalonadores comparados, neste trabalho, porque ele privilegia as classes mais prioritárias, mas além disso, redistribui os recursos de maneira a não permitir que o desempenho das outras classes caia.

Como trabalho futuro, sugere-se a imlementação de um algoritmo genético, que leve em consideração o estado final das simulações como função objetivo. Ele pode ser utilizado para melhorar a escolha dos parâmetros que definem os níveis de pertinência de cada uma das variáveis. A utilização de modelos de mobilidade que reflitam mais a movimentação de usuários comuns como, por exemplo, usuários em automóveis e transporte público e simulações com uma maior aleatoriedade na duração de cada tipo de chamada e a utilização de outros mecanismos de garantia de qualidade de serviço em conjunto com o escalonador como controles de admissão de chamadas, também são trabalhos futuros.

Agradecimentos

Agradecemos à CAPES e à Microsoft Research, sem as quais este trabalho não poderia ter sido executado.

Referências

3GPP (2013). Lte release 10. http://www.3gpp.org/technologies/keywords-acronyms/97-lte-advanced. Acessado em 10.04.15.

Andrews, M., Kumaran, K., Ramanan, K., Stolyar, A., Whiting, P., and Vijayakumar, R. (2001). Providing quality of service over a shared wireless link. *IEEE Communications Magazine*, 39(2):150–154.

Ball, C., Treml, F., Gaube, X., and Klein, A. (2005). Performance analysis of temporary removal scheduling applied to mobile wimax scenarios in tight frequency reuse. In *Personal, Indoor and Mobile Radio Communications, 2005. PIMRC 2005. IEEE 16th International Symposium on*, volume 2, pages 888–894 Vol. 2.

de Souza, F. R. (2016). Fuzsy: um escalonador de pacotes baseado em qualidade de serviço e lógica fuzzy. Master's thesis, Pontifícia Universidade Católica de Minas Gerais.

IBGE (2016). Projeções e estimativas da população do brasil e das unidades da federação. Disponível em: <http://www.ibge.gov.br/apps/populacao/projecao/notatecnica.html>.

Jalali, A., Padovani, R., and Pankaj, R. (2000). ata throughput of cdma-hdr a high efficiency-high data rate personal communication wireless system. In *Vehicular Technology Conference Proceedings, 2000. VTC 2000-Spring Tokyo. 2000 IEEE 51st*, volume 3, pages 1854–1858 vol.3.

Lai, W. K. and Tang, C.-L. (2013). Qos-aware downlink packet scheduling for {LTE} networks. *Computer Networks*, 57(7):1689 – 1698.

Lai, Y.-L. and Jiang, J.-R. (2014). Pricing resources in lte networks through multiobjective optimization. volume 2014, page 8.

Piro, G., Grieco, L., Boggia, G., Capozzi, F., and Camarda, P. (2011a). Simulating lte cellular systems: An open-source framework. *Vehicular Technology, IEEE Transactions on*, 60(2):498–513.

Piro, G., Grieco, L. A., Boggia, G., Fortuna, R., and Camarda, P. (2011b). Two-level downlink scheduling for real-time multimedia services in lte networks. *IEEE Transactions on Multimedia*, 13(5):1052–1065.

Pizzi, S., Molinaro, A., and Iera, A. (2009). On the performance of "compensation-based"and "greedy"scheduling policies in ieee 802.16 networks. In *Communications, 2009. ICC '09. IEEE International Conference on*, pages 1–6.

Rhee, J.-H., Holtzman, J. M., and Kim, D. K. (2004). Performance analysis of the adaptive exp/pf channel scheduler in an amc/tdm system. *IEEE Communications Letters*, 8(8):497–499.

Sadiq, B., Baek, S. J., and de Veciana, G. (2011). Delay-optimal opportunistic scheduling and approximations: The log rule. *Networking, IEEE/ACM Transactions on*, 19(2):405–418.

Shakkottai, S. and Stolyar, A. L. (2002). Scheduling for multiple flows sharing a time-varying channel: The exponential rule. *Translations of the American Mathematical Society-Series 2*, 207:185–202.

Tarchi, D., Fantacci, R., and Bardazzi, M. (2006). Quality of service management in ieee 802.16 wireless metropolitan area networks. In *Communications, 2006. ICC '06. IEEE International Conference on*, volume 4, pages 1789–1794.

Tostes Ribeiro, A. I. J., Zarate, L. E., and Duarte Figueiredo, F. d. L. P. (2013). Dynamic fuzzy cellular admission control. In *5th IEEE Latin-American Conference on Communications (IEEE LATINCOM 2013)*.

Wallenius, E. and Hamalainen, T. (2002). Pricing model for 3g/4g networks. In *Personal, Indoor and Mobile Radio Communications, 2002. The 13th IEEE International Symposium on*, volume 1, pages 187–191 vol.1.