O uso de uma máquina de estados finitos para modelar um protocolo

David Teixeira de Masin, Christopher Breno Coelho Xavier, José Fernando de Almeida Teobaldo Júnior

Instituto Federal de Educação, Ciência e Tecnologia do Ceará - IFCE
Graduando em Tecnologia em Telemática
Avenida 13 de Maio, 2081, Benfica
Fortaleza/CE, Brasil, 60040-531
{divield}@gmail.com, {christopherbreno}@gmail.com, {jfernandoalmeidatj}@gmail.com

Resumo – Este artigo apresenta uma forma de formalizar e compreender melhor o comportamento de um protocolo de comunicação de redes seja ele, unicast ou multicast, através da utilização de uma máquina de estados finitos. Para tal finalidade, será adotado um software específico, que funciona em qualquer sistema operacional, mas, será necessário ter o Java instalado no computador a ser manuseada, para a construção e simulação de uma máquina de estados.

Abstract – This article presents a way to formalize and better understand the behavior of a communication protocol networks be it, unicast or multicast, through the use of a finite state machine. For this purpose, will be adopted a specific software which works on any operating system, but you must have Java installed on the computer to be handled, for the construction and simulation of o state machine.

I. INTRODUÇÃO

No mundo das Redes de Computadores, existe a possibilidade da aplicação do conceito de máquina de estados, que se trata de um modelo matemático utilizado na representação de programas ou circuitos lógicos, para prover uma melhor visão do comportamento de um determinado protocolo de comunicação, seja ele *unicast* ou *multicast*, e possibilitando criar um modelo formal do mesmo. Este conceito permite ainda, a capacidade de qualquer pessoa projetar o seu próprio protocolo.

Modelar um protocolo de comunicação através de uma máquina de estado não é algo fácil, no decorrer dos anos a tecnologia foi melhorada e isso resultou em certo nível de complexidade, consequentemente, isso influenciou no formato dos protocolos. [1]

A máquina de estados a ser fundamentada neste artigo é a de estados finitos, conhecido pela sigla em inglês *FSM* (*Finite State Machine*) ou pela sigla em português MEF (Máquina de Estados Finitos). Ela é bastante utilizada na modelagem de diversos problemas tais como, a automação de design eletrônico, projeto de protocolo de comunicação, análise, entre outras situações. [6]

Uma máquina de estados finitos, como o nome já sugere, possui uma quantidade finita de estados que ocorre um por vez e quando um deles está ativo, recebe o nome de estado atual. Além dos estados, uma *FSM* possui transições, que se referem às mudanças de estado, onde são compostas de eventos e ações. O evento representa uma situação que irá permitir a ocorrência da transição e a ação nada mais é que a consequência desse evento.

Na fig. 1 a seguir, temos um exemplo bem simples de uma máquina de estados finitos representando a situação da catraca do ônibus. Para o estado "TRAVADA" sofrer uma transição para o "DESTRAVADA", é necessário que ocorra o evento "Pagar passagem" que resulta na ação "Destrava". Quando no estado "DESTRAVADA" para retornar à "TRAVADA" deve ocorrer o evento "Passando pela catraca" que gera uma ação "Trava".

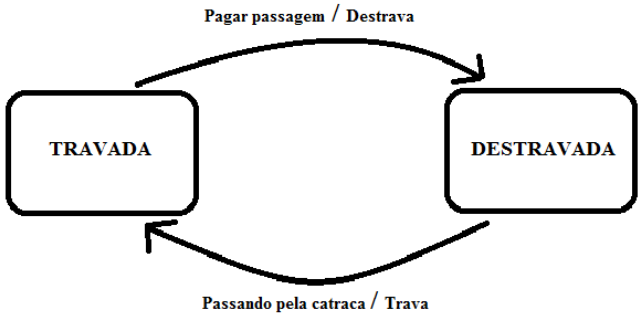

Fig. 1 - Uma máquina de estados finitos simples representando o comportamento da catraca do ônibus.

II. SOFTWARE PARA O DESENVOLVIMENTO DE UMA FSM

Para o desenvolvimento de uma *FSM*, foi adotada a utilização do software *Automaton Simulator* ou simplesmente *AutoSim*, que trata-se de um aplicativo desenvolvido em Java que tem a capacidade de projetar diversos tipos de máquinas de estados. O computador que fará o uso deste aplicativo deverá ter obrigatoriamente o Java, versão 1.3 ou superior, instalado para ser possível a execução do *AutoSim*.[4]

De forma breve, o *Automaton Simulator* é um software livre capaz de ser executado em qualquer sistema operacional, desde que tenha o Java instalado, que nos permite construir e simular uma variedade de máquinas teóricas, tais como, máquinas de estados finitos determinísticos, finitos não-determinísticos, determinísticos *push-down* e máquinas de *Turing*. [2] [3] [5] Na fig. 2 a seguir, será possível visualizar a aparência deste software e conhecer a construção de uma *FSM* qualquer.

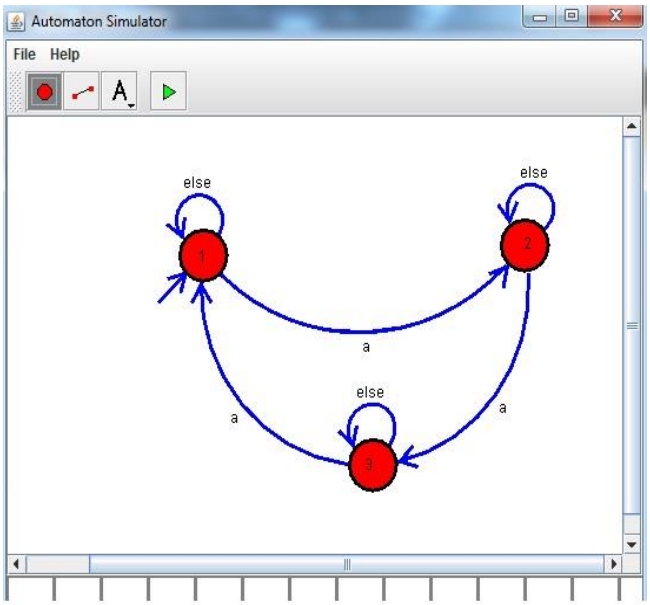

Fig. 2 - Uma visão geral do *Automaton Simulator* com uma FSM qualquer projetada.

Na fig. 2, os estados estão representados pelos círculos preenchidos de vermelho, quando um destes estiver verde, significa que é o estado atual. Dando continuidade, as setas representam as transições e elas podem ser definidas pelas letras ‘a’, ‘b’, ‘c’ ou ‘d’, ou por ‘*else*’. O estado 1 possui uma seta a mais em relação ao demais estados e ela indica o início, ou seja, é por onde a máquina de estados irá começar.

Baseando-se ainda pela fig. 1, se a letra ‘a’ do teclado for pressionada duas vezes, teremos a transição do estado 1 para o 2 e do 2 para o 3. Se for pressionada outra tecla diferente do ‘a’ quando um dos círculos estiver preenchido de verde, não ocorrerá mudança de estado e permanecerá no mesmo. Quando temos uma situação que não é prevista em um determinado estado de uma *FSM*, é definida uma transição *else* para evitar que a máquina de estados fique travada. Seria bastante viável implementar essa transição durante a modelagem de um protocolo pois, iria prevenir de qualquer situação não prevista em um determinado estado.

No tópico seguinte, será contextualizada a modelagem de um protocolo *unicast* e *multicas*t.

III – MODELANDO PROTOCOLOS UNICAST E MULTICAT

Nas redes *TCP/IP* temos protocolos *unicast* e *multicast*. Os protocolos unicast são caracterizados por serem capazes de gerar uma comunicação única entre dois processos e os protocolos multicast caracterizam-se pela capacidade de um processo poder transmitir uma mesma mensagem para uma quantidade N de destinatários.

Vamos partir para a modelagem dos protocolos unicast, como exemplos temos, o *PPP*, o *FTP*, o *Telnet*, o *SMTP*, etc. Dos citados anteriormente, faremos uso do protocolo *PPP*, antes de modelá-lo é importante que se tenha uma breve visão dele para facilitar no desenvolvimento de sua máquina de estados.

O protocolo *PPP* (*Point to Point Protocol*) é aplicado para estabelecer uma conexão direta entre dois nós na rede. Ele está apto a fornecer a autenticação da conexão, criptografia da transmissão e compressão. [7]

Uma *FSM* que define o comportamento do protocolo *PPP* apresentará 5 estados e eles são: enlace morto, estabelecimento do enlace, autenticação do enlace, rede e terminado. Na fig. 3, temos a máquina de estados desenvolvido no *AutoSim* que simula o comportamento do *PPP*.

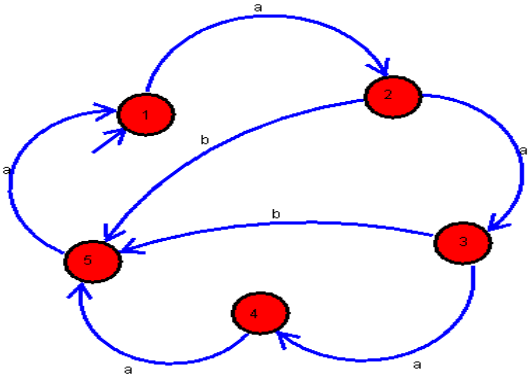

Fig. 3 - Representação da máquina de estados finitos do PPP.

Na *FSM* deste protocolo *unicast*, os estados de 1 a 5 são, respectivamente, enlace morto, estabelecimento do enlace, autenticação do enlace, rede e terminado. Na tabela I será possível verificar a especificação dos estados da *FSM* projetada e na tabela II, a descrição das transições.

TABELA I - ESPECIFICAÇÃO DOS ESTADOS DO *PPP* [7]

Estado	Nome	Descrição
1	Enlace Morto	Ausência de conexão.
2	Estabelecimento do Enlace	Conexão estabelecida, mas, precisa ser autenticada.
3	Autenticação do Enlace	Autenticação ocorre quando entra com usuário e senha.
4	Rede	Ocorrência da comunicação de dados.
5	Terminado	É enviada a solicitação de desconexão.

TABELA II - DESCRIÇÃO DAS TRANSIÇÕES [7]

Transição	Descrição
a	Transição do estado 1 para 2, constituída pela ação "Enlace Liga".
a	Transição do estado 2 para 3, constituída pela ação "Abre Conexão".
a	Transição do estado 3 para 4, constituída pela ação "Usuário e Senha corretos".
a	Transição do estado 4 para 5, constituída pela ação "Fecha Conexão".
a	Transição do estado 5 para 1, constituída pela ação "Conexão".
b	Transição do estado 2 para 5, constituída pela ação "Falha".
b	Transição do estado 3 para 5, constituída pela ação "Falha".

Após modelarmos um protocolo *unicast*, faremos o mesmo com um protocolo *multicast*. Temos como exemplos, os protocolos *IGMP*, *RTP*, *IRC*, etc. Para a modelagem, será adotado o protocolo *IGM*P.

De forma breve, o *IGMP* (*Internet Group Management Protocol*) é um protocolo utilizado pelos hosts e roteadores adjacentes nas redes *IP* com o objetivo de estabelecer os membros de um grupo *multicast*, sendo ainda parte do *IP multicast*. Pode ser implementado em aplicações do tipo streaming, permitindo que a sua utilização seja mais eficiente. [7]

Serão utilizados três estados para a construção da máquina de estados do *IGMP* e eles são: não membro, membro, aguardando para se tornar membro. Na fig. 4 é possível verificar a *FSM* projetada.

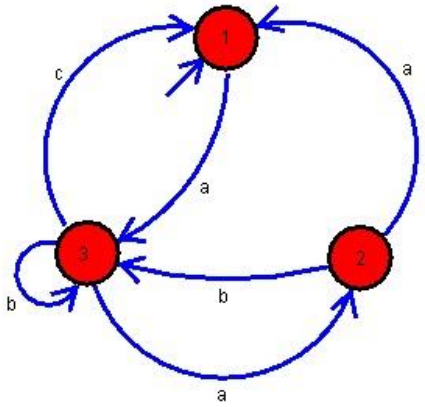

Fig. 4 - Máquina de estados finitos do protocolo IGMP.

A tabela III a seguir, apresenta a descrição dos estados que constituem a *FSM* da fig. 4.

TABELA III - DESCRIÇÃO DOS ESTADO DO *IGMP*[7]

Estado	Nome	Descrição
1	Não Membro	Determina uma estação como não membro de nenhum grupo *Multicast*.
2	Membro	Reconhece uma estação com membro de um grupo *Multicas*t.
3	Aguardando para se tornar Membro	A estação recebe uma mensagem Query do roteador para sabe se faz parte de algum grupo *Multicast*.

Com uma boa assimilação da descrição de cada estado, vamos entender o que cada transição representa com base na tabela IV.

TABELA IV - DESCRIÇÃO DAS TRANSIÇÕES [7]

Transição	Descrição
a	Transição do estado 1 para 3, chama-se "Entra grupo". Define que uma estação deseja entra no grupo Multicast.
a	Transição do estado 2 para 1, chama-se "Deixa Grupo". Define que a estação abandona o grupo que anteriormente fazia parte.
a	Transição do estado 3 para 2, chama-se "Report Recebido ou Tempo Expirado". Define a volta da estação ao estado 1 quando recebido a mensagem report.
b	Transição do estado 2 para 3, chama-se "Query Recebida". Define o retorna da estação ao estado 3.
b	Transição que permaneça no estado 3, chama-se "Query Recebida". Caso seja recebido no estado 3 outra mensagem query nada irá ocorrer.
c	Transição do estado 3 para 1, chama-se "Deixa Grupo". Define que a estação abando o grupo *multicast* o qual fazia parte.

IV - POSSIBILIDADE DE PROJETAR O SEU PRÓPRIO PROTOCOLO DE COMUNICAÇÃO

No tópico anterior, verificamos a capacidade de criar modelos formais de protocolos, seja ele *unicast* ou multicast, através de uma máquina de estados finitos.

O software *Automaton Simulator* é simples, de fácil manuseio e mostra-se bem útil na modelagem de um protocolo de comunicação. Permite ainda, que qualquer pessoa possa, a partir desse software, projetar o seu próprio protocolo.

V - CONCLUSÃO

O artigo expõe uma visão de como funciona a modelagem de um protocolo de comunicação por meio de uma máquina estados finitos com a utilização de uma aplicação em Java, o *AutoSim*.

Houve a oportunidade de obter conhecimento acerca do conceito de máquina de estados com ênfase na de estados finitos. Teve espaço ainda para uma abordagem do software utilizado para a modelagem, informando algumas especificações e requisito para uma correta utilização.

Por fim, foi mostrado como se projeta um protocolo e citado a possibilidade de qualquer pessoa projetar o seu.

REFERÊNCIAS

[1]http://www.bcc.ufla.br/monografias/2001/Modelagem_e_implementacao_de_protocolos_para_comunicacao_com_e_sem_fio_de_um_sistema_integrado_hardware_software_em_tempo_real.pdf. Em 27/09/13, 22:30h.
[2] http://ozark.hendrix.edu/~burch/proj/autosim/. Em 29/09/13, 11:30h.
[3]http://ozark.hendrix.edu/~burch/proj/autosim/doc/about.html. Em 29/09/13, 11:32h.
[4]http://ozark.hendrix.edu/~burch/proj/autosim/download.html. Em 29/09/13, 11:35h.
[5]http://ozark.hendrix.edu/~burch/proj/autosim/history.html. Em 29/09/13, 11:34h.
[6]http://pt.wikipedia.org/wiki/M%C3%A1quina_de_estados_finitos#Diagrama_de_estados. Em 03/10/13, 00:15h.
[7]http://virtual.ifce.edu.br/moodle/file.php/1/Etec/Redes_de_Computadores/2013.2/Semestre_II/Protocolos_de_Comunicacao/ETEC_-_IFCEApostila.pdf. Em 02/10/13, 01:55h.