Estudo da API framework Crossbar.IO e suas aplicações em sistemas robóticos

Gabriel João Schiller (Fundação Araucária), Ariangelo Hauer Dias (Orientador), gabrielschiller8@gmail.com:ariangelo@gmail.com.

Universidade Estadual de Ponta Grossa/Departamento de Informática/ Setor de Ciências Agrárias e de Tecnologia.

Engenharias; Engenharia Elétrica; Eletrônica Industrial, Sistemas e Controles Eletrônicos

Palavras-chave: robótica, iot, framework

Resumo

Na atualidade, muito se fala sobre desenvolvimento da sociedade por moldes de automação. Com o uso de robôs, há a necessidade de controle para tais. A API framework Crossbar.IO nos fornece esse controle e se torna uma ótima escolha para uso por ser um projeto de código aberto, abrangente e em constante desenvolvimento. Sendo essa API multi linguagem e em constante crescimento, observamos sua viabilidade em tratar projetos de robótica e sua abordagem em websocket faz a conectividade dos componentes excepcional.

Introdução

O uso da robótica em meio residencial tem se tornado um meio essencial da vida humana. É de se esperar que no futuro esse uso se torne comum como necessário, tornando essa uma área de pesquisa abrangente e requisitada.
Com o uso de robôs, há a necessidade de controle para tais. Esse controle é feito através da interface entre o físico e o virtual, ou seja, hardware e software, conectando esses para um fácil manuseio, portabilidade e sempre maximizando o uso (1). O API framework crossbar.IO nos fornece tal controle com abrangência, em virtude de seu desenvolvimento conceder muitas funcionalidades e por ser um framework de código aberto, podendo assim modificar ao seu uso.

Desta forma, este trabalho tem como objetivo avaliar a funcionalidade dos sistemas de distribuição de informações por micro mensagens Crossbar.IO e sua aplicação em sistemas robóticos.

Materiais e Métodos

Para se utilizar a API framework Crossbar.io deve-se entender alguns conceitos que abrangem a ferramenta. Uma API, interface de programação de aplicações (Application Programming Interface), é um conjunto de procedimentos padronizados por um software que se desprende de detalhes apresentados na implementação do software e se foca apenas no uso de seus serviços (2). A API Crossbar.io tem como ideia principal *WAMP*.

Não se deve confundir *WAMP* com o acrônimo para a combinação de *Windows*, *Apache*, *My SQL*, *PHP-Perl-Python*, pois esse é um pacote disponibilizado para a plataforma Windows para desenvolvimento de aplicações Web.

O *WAMP* a qual é referido ao *Crossbar.IO* é um protocolo de comunicação usando a tecnologia de *Websockets*. Esse nome deriva de *Web Application Messaging Protocol*, que em tradução significa Protocolo de mensagens de aplicação web (3). Utilizando a biblioteca *Autobahn*, ele consegue comunicar-se entre diversas plataformas, facilitando a troca de informações. O conceito básico para essa comunicação vem da implementação de procedimentos *Publish and Subscribe (PubSub)* e chamadas de procedimento remoto em várias plataformas de programação (4).

A ideia baseia-se na troca de informação entre o método *Publish* e *Subscribe*. Qualquer cliente que tenha acesso ao servidor pode se inscrever em um tópico de troca de mensagens. Esse tópico é criado pelo desenvolvedor tal como se fosse uma variável nos termos de programação. Quando um cliente está inscrito em um tópico ele pode enviar informações através do método *Publish*, e todo cliente inscrito no tópico a qual ele está enviando as informações iram receber, incluindo ele mesmo. Desse modo a troca de informações é feita multiplataformas.

Seguindo o mesmo conceito de *Publish* e *Subscribe*, temos o procedimento de chamada de procedimento remoto. Dentro da biblioteca *Autobahn* é declarada como *call*. Um cliente pode registrar um procedimento à um tópico pelo procedimento *register* e esse é disponibilizado na malha do *WAMP*. Desse modo, qualquer cliente pode invoca-lo através de um simples *call*.

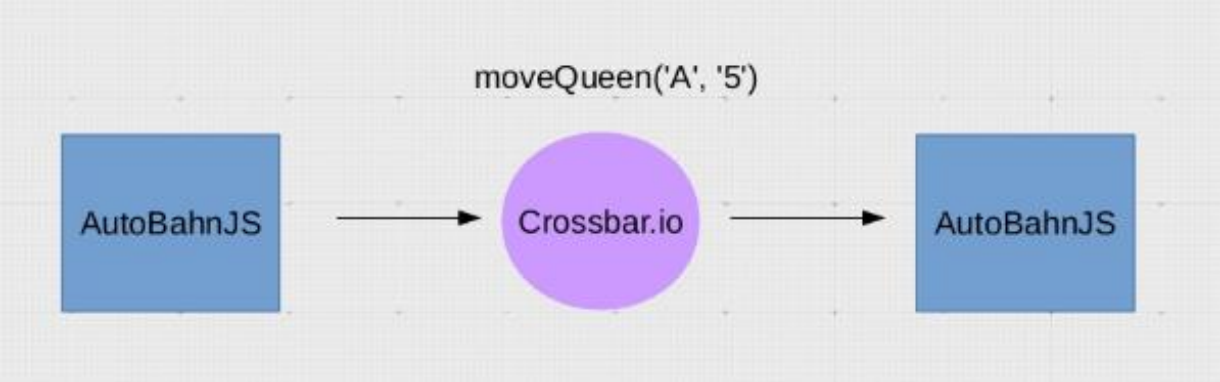


Figura 1 – Representação de uma chamada call em um jogo de xadrez utilizando a biblioteca AutobahnJS

Na Figura 1 vemos um exemplo de uma comunicação. Percebemos que há a necessidade de um server central para a comunicação dos clientes já que a comunicação não é P2P (Peer-to-peer, arquitetura de rede que cada nódulo conectado à estrutura é cliente e server), sendo esse server central de roteamento o crossbar.IO. O cliente AutobahnJS a esquerda invoca o procedimento moveQueen('A','5). Esse procedimento é então enviado ao crossbar.IO e então redirecionado ao cliente AutobahnJS a direita para esse executar a chamada.

É de se notar que as linguagens suportadas pela biblioteca *Autobahn* até o presente momento são: *Python 2* e *3*, *JavaScript*, *C++*, *C* e *Java*.

As configurações do servidor *Crossbar.IO* são feitas em um arquivo formato JSON (Java Script Object Notation). JSON é uma formatação de troca de dados, tendo em vista que em quase todas as linguagens ela é suportada, por essa razão ela é utilizada como arquivo de configuração do crossbar.IO, podendo ser carregada em várias linguagens (5). Essa estrutura é apenas uma lista de objetos seguidas de nome/valores.

No arquivo JSON devemos ressaltar alguns pontos. Toda a configuração com o Crossbar.IO é feita através de realms. Um servidor pode ter vários realms, e estes podem ser configurados de maneira independentes. Realms funcionam como sub redes a qual podemos conectar nossos clientes. Esses realms são referidos como variáveis a qual o desenvolvedor pode chama-la a seu grado. Podendo também configurar esses realms com acesso restringido para as chamadas roles. Roles é o nome dado aos clientes que acessam o sistema. Podemos definir o que cada roles pode fazer com base no conceito do WAMP. Desse modo, podemos limitar uma role a apenas chamar procedimentos, registrar a tópicos, publicar em tópico ou se inscrever a um. Genericamente atribuímos por padrão uma role *anonymous* como qualquer cliente no sistema.

Resultados e Discussão

Nesses meses de pesquisa foram desenvolvidas aplicações para o estudo da ferramenta API framework Crossbar.IO. Como a base de um sistema robótico é o recebimento e tratamento de dados tais como a manipulação, as aplicações não foram feitas com o intuito de testar o físico de sistemas robóticos.

Os resultados observados foram obtidos no tratamento de informações enviadas de um microcontrolador.

A leitura de um potenciômetro conectada a uma entrada do microcontrolador era enviada ao computador por meio de passagem de informação Serial. Essa leitura então era enviada ao servidor *Crossbar* instalada no mesmo computador. Com a utilização de JavaScript, o dado era atualizado na tela de forma a não observar incoerências visíveis aos olhos conforme o potenciômetro era modificado, pois no servidor tínhamos uma escala de 0 a 100% do nível do potenciômetro.

Uma replica da aplicação foi feita e aplicada em outro *realm* onde a demonstração da escala do potenciômetro era feita de maneira inversa, ou seja, de 100 a 0%. Nota-se que a primeira aplicação não foi deletada e estava funcionando de maneira paralela com a segunda. Os dois dados eram atualizados em tempo real como já explicado anteriormente, sem demonstração de incoerência com o estado real.

Os dados foram todos tratados em um servidor. A latência no recebimento de informações foi proporcional a um servidor *LAN* local, sendo essa latência apta a serviços de resposta ditas quase imediata.

É de se esperar que em servidores a distância essa latência esteja relacionada ao trafego dos dados, dessa forma, torna o crossbar.IO de extrema utilidade para serviços de automação, tais como automação residencial e seu acompanhamento por serviços *web*.

Conclusões

Com base em todo o processo de pesquisa envolvendo a API framework Crossbar.IO vemos que ele suporta projetos de robótica. Sua característica de subdivisão das conexões em *Realms*, faz de bom uso caracterizações de vários dispositivos conectados a ela, podendo analisar e ter consciência do que cada dispositivo pode ou não fazer, como o que ele está fazendo em poucas linhas de configuração. É importante ressaltar que o termo dispositivo é referido a todo equipamento no sistema, sendo eles microcontroladores dos equipamentos

atuadores, como também clientes. Esse modo de divisão também nos traz a possibilidade de termos vários sistemas conectados a um centro, diminuindo assim a complexidade de várias aplicações.

Sua característica de multi linguagem também traz uma ótima vantagem à plataforma. Desse modo, podemos invocar procedimentos de outros lugares, programados em outras linguagens de maneira simples e funcional. A facilidade em programar determinadas áreas do sistema em determinadas linguagens nos leva à tona a real importância dessa característica.

Considerando os resultados obtidos é possível concluir que o Crossbar.IO é uma ferramenta adequada a troca de informações de sistemas robóticos utilizando como meio de troca a internet.

Agradecimentos

Agradeço a Universidade Estadual de Ponta Grossa pela oportunidade de nos prestigiar com projetos voltados ao rumo cientifico. Agradeço também a fundação Araucária por patrocinar esses meses de pesquisa, e aos professores Ariangelo Hauer Dias e Dierone Cesar Foltran Junior pela possibilidade de pesquisa e orientações.

Referências

1. CRAIG, J. J. Introduction to Robotics: Mechanics and Control. 3. ed. 1989, p. 408.
2. Application Programming Interface. Disponível em: <https://www.computerworld.com/article/2593623/app-development/application-programming-interface.html>. Acesso em 22 de Setembro de 2018.
3. A quick introduction to WAMP. Disponível em: <https://wamp-proto.org>. Acesso em 27 de Agosto de 2018.
4. What is crossbar.io. Disponível em <http://crossbar.io/about/> Acesso em 27 de Agosto de 2018.
5. Introdução ao JSON. Disponível em <https://www.json.org/json-pt.html>. Acesso em 27 de Agosto de 2018.