UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
DEPARTAMENTO DE ELETRÔNICA E COMPUTAÇÃO

SISTEMAS OPERACIONAIS DE TEMPO REAL
ELEVADOR INTELIGENTE

IURY CLEVESTON

PROFESSOR: OSMAR MARCHI DOS SANTOS

SANTA MARIA - RS
15 DE JUNHO DE 2015

INTRODUÇÃO

A maioria dos sistemas que controlam o funcionamento dos elevadores hoje em dia são desprovidos de métodos de otimização de deslocamento. Isto é, o elevador opera em modo FIFO (First in, First out), ele executa as requisições conforme estas vão chegando, sem nenhuma análise matemática para otimizar suas tarefas.

Dito isso, o presente trabalho tem por objetivo explicar a implementação e funcionamento de um elevador inteligente.

Definimos como sistema inteligente, nesse caso, o sistema que gera o melhor caminho que atenda as requisições dos usuários. A melhor solução para esse problema é permutar todos os andares requisitados e somar os deslocamentos para cada trajeto, de modo a encontrar o trajeto com o menor deslocamento total final.

Sabemos que este é um problema de otimização NP Difícil chamado “Problema do Caixeiro Viajante”, cuja complexidade da solução cresce de forma exponencial conforme o número de andares e requisições aumentam. Este método requer muito poder computacional, tornando inviável para um sistema embarcado.

Nosso sistema implementa uma solução em grafos, só que sem consultar todos os possíveis trajetos, ele executa a melhor requisição a partir do andar onde o elevador se encontra parado.

Para isso, este trabalho foi implementado na linguagem de programação C, utilizando Socket para a comunicação entre os módulos, Threads para concorrência das tarefas e Tarefas Periódicas, além de uma biblioteca para a manipulação de grafos Open Source. A seguir explicamos como foi feita a implementação.

DESCRIÇÃO DA IMPLEMENTAÇÃO

Antes de iniciarmos nossa implementação devemos fazer uma pequena modificação no nosso elevador: tiraremos o painel de comandos de dentro do elevador e o colocaremos em cada andar.

Desse modo, o usuário escolhe previamente o andar que quer ir antes do elevador chegar, com essa modificação conseguimos otimizar o deslocamento.

Sendo assim, o projeto consiste em dois módulos: Elevador e Andar. O módulo Elevador deve ser executado primeiro, o administrador deve fornecer o número de andares que o prédio possui.

E a partir de então, o módulo Andar deve ser executado para cada terminal em cada andar. Quando tivermos o número de andares suficientes conectados ao elevador, este entra em funcionamento e para de receber conexões de andares, por motivos de segurança.

O elevador responde as requisições dos usuários, isto é, quando algum usuário deseja ir para outro andar, o sistema monta um grafo de onde o usuário se encontra e para onde quer ir. Caso mais requisições ocorram nesse mesmo tempo, elas são adicionadas ao grafo. Com o grafo montado o elevador poderá decidir qual usuário atenderá primeiro.

Para isso, o menu que está disponível para os usuários consiste nos seguintes comandos:

- IR ‘andar’
- AJUDA
- EMERGENCIA
- ATIVAR (acessado apenas pelo administrador do prédio)
- DESATIVAR (acessado apenas pelo administrador do prédio)

O comando IR ‘andar’ informa ao elevador qual andar o usuário quer ir. Quando isso acontece, a requisição é adicionada ao grafo.

O comando AJUDA mostra um menu com os comandos disponíveis.

O comando EMERGENCIA faz com que a requisição seja passada a frente das demais. Quando isso ocorre, o elevador vai diretamente para o andar informado e leva o passageiro ao térreo.

O comando ATIVAR e DESATIVAR faz com que o andar seja ativado ou desativado pelo administrador do prédio, quando isso ocorre, não é possível ir para este andar.

Sendo assim, os andares são conectados ao elevador através de Sockets em C. E cada andar é tratado por uma thread, ou seja, nosso sistema pode atender diferentes requisições simultaneamente. Ademais, este sistema não tem limites de andares ou de requisições, pois utiliza alocação dinâmica de memória.

Quando o elevador chega a um determinado andar, ele envia o comando ‘abrePorta’, informando ao andar que ele está pronto para a entrada dos passageiros. Quando os passageiros entrarem, o elevador envia o comando ‘fechaPorta’ para o andar e inicia seu deslocamento.

Caso aconteça alguma falha na comunicação com algum andar, por medidas de segurança, o elevador vai para o térreo e aguarda o andar se conectar novamente, através de um rotina periódica.

Cada requisição dos usuários é gravada num arquivo de log, juntamente com a hora.

CONCLUSÃO

O sistema foi construído usando grafos com a intenção, para uma versão futura, de implementar um algoritmo mais eficiente na otimização do deslocamento. É possível que os algoritmos para resolução de Asymmetric TSP nos dêem bons resultados, pelo menos para um número limitado de andares e de fluxo de usuários.

Outra melhoria consiste em determinar, através de sensores, o número de passageiros presentes dentro do elevador e realizar a lógica com base nessa variável, para que o elevador não busque mais passageiros quando se atingiu a capacidade máxima.

Usando um algoritmo que gera todo o deslocamento previamente, é possível colocar um display em cada andar, informando quanto tempo falta para o elevador chegar. Isto é, quando o usuário aperta o botão do andar que deseja ir, ele já visualiza quanto tempo falta pra o elevador chegar até ele.