P4: Spatial synchronization

Ad Nunes Ribeiro (11200620), Eduardo Dias Defreyn (11203780),
Ricardo do Nascimento Boing (14200760)
Bacharelado em Ciência da Computação -- Universidade Federal de Santa Catarina (UFSC)
Campus Reitor João David Ferreira Lima, 88.040-900 -- Florianópolis -- SC -- Brasil

adnunesribeiro@gmail.com; eduardo_dududex@hotmail.com; ricardoboing.ufsc@gmail.com

1. Spatial synchronization

O grupo desenvolveu e implementou as alterações necessárias ao P3 para também tratar a sincronização espacial. Para isso foram adicionados dois novos campos ao pacote do protocolo: o ponto que representa a localização do nodo, e se este ponto é válido.

Para o cálculo de posicionamento foi utilizada a trilateração 2D mas todo o protocolo foi desenvolvido sobre pontos XYZ,utilizando o template Point da EPOS, de modo que a trilateração pode ser alterada sem a necessidade de alterar o protocolo.

Devido a diferenciação entre a classe SOS e SOS_Communicator a sincronização espacial, assim como a temporal, está ligada a classe SOS, sendo independente de porta. Para a demonstração foi definido um nodo que tem acesso a serial de onde são obtidos os dados das âncoras e enviados para o nodo que necessita ser sincronizado. O nodo que necessita de sincronização por sua vez, recebe dados distintos de três âncoras(considerando a trilateração 2D) e calcula a posição do nodo.

2. Primeira Sincronização

Se uma instância de SOS tentar fazer o envio de um pacote, será verificada se a localização atual é válida. Caso não seja, antes de realizar o envio do pacote para o endereço destino serão solicitados dados de localização das âncoras até ser feito o cálculo da trilateração. Como o cálculo ocorre na classe SOS não afeta o recebimento de dados na aplicação.

3. Cálculo de distância

Para simular o cálculo da distância, devida às limitações da VM, foi adicionado um ponto de referência a classe SOS que representa o ponto esperado para o nodo. Deste modo quando um nodo necessita saber a distancia de uma âncora só é necessário fazer o cálculo da distância euclidiana entre o ponto recebido e o ponto esperado.

4. Âncoras

são representados pela estrutura que guarda a validade da ancora, sua localização e distância e tem seus valores preenchidos no recebimento de pacotes.

5. Classe GPS_Driver

Sobre a obtenção dos dados através da serial, foi criada a classe GPS_driver responsável por realizar a conexão com o host-side, através de uma uart, além de fazer o tratamento dados dados recebidos até o formato esperado pelo protocolo(XYZ).

6. aplicação GPS host-side

A aplicação host-side se encontra na pasta sos_tests com o nome client.py e representa uma simples aplicação que recebe um número através da serial e retorna o valor correspondente em um array.

7. dificuldades encontradas

As maiores dificuldades encontradas estão relacionadas a manipulação de números, inicialmente foi descoberto que tipos de ponto flutuante causam referência para void no template Point, por isso foi decidido utilizar valores inteiros, ainda que se tenha uma perda de precisão.

Na obtenção dos valores numéricos obtidos pela serial acontece o mesmo problema. Considerando que a classe string.h não apresenta conversão para ponto flutuante, o método manual para o mesmo apresenta imprecisão. Tendo então os valores de LLA, mesmo que com um leve erro, os cálculos providos no seminário utilizam seno e cosseno, métodos que não estão disponíveis em math.h, ainda que possa ser feita uma implementação da expansão da série de taylor isso causa novamente a perda de precisão.

Obtendo os valores de seno e cosseno por aproximação(com erro) ainda ocorre um erro no método sqrt com valores de ponto flutuante.