Санкт-Петербургский политехнический университет Петра Великого
Высшая школа прикладной математики и вычислительной физики
Кафедра прикладной математики

Курсовая работа
по дисциплине «Компьютерные сети»
на тему

Задача византийских генералов

Выполнил студент гр. 5040102/00201
Чепулис М.А.

Преподаватель
Баженов А.Н.

Санкт-Петербург
2021 год

Оглавление

Постановка задачи

Задача византийских генералов:

В [1] описана следующая постановка задачи византийских генералов.

Пусть n «белых» генералов возглавляют армии в горах и готовятся атаковать «чёрных» в долине. Для связи атакующие используют надёжный канал (например, телефон), исключающий подмену сказанного. Однако из n генералов m являются предателями и активно пытаются воспрепятствовать согласию лояльных генералов. Согласие состоит в том, чтобы все лояльные генералы узнали о численности всех лояльных армий и пришли к одинаковым выводам (пусть и ложным) относительно состояния предательских армий. (Последнее условие важно, если генералы на основании полученных данных планируют выработать стратегию, и необходимо, чтобы все генералы выработали одинаковую стратегию.)

По результатам обмена каждый из лояльных генералов должен получить вектор целых чисел длины n, в котором i-й элемент либо равен истинной численности i-й армии (если её генерал лоялен), либо содержит дезинформацию о численности i-й армии (если её генерал не лоялен). При этом векторы, полученные всеми лояльными командирами, должны быть полностью одинаковы.

Необходимо обеспечить протоколы общения между главнокомандующими и реализовать алгоритм решения задачи византийских генералов.

Реализация

Описание алгоритма

Алгоритм решения задачи византийских генералов:

В нашем случае количество неверных генералов не изменяется со временем. Для такой постановки в 1982 году Лесли Лампорт предложил рекурсивный алгоритм, который задачу для случая m предателей генералов сводит к случаю $m-1$ предателя.

Определение алгоритма Лэмпорта [2]

Алгоритм $OM(0)$

1. Генерал посылает каждому лейтенанту своё значение
2. Каждый лейтенант использует значение, которое получает от генерала.

Алгоритм $OM(m)\ m > 0$

1. Генерал посылает каждому лейтенанту своё значение
2. Для каждого i пусть v_i будет значением, которое лейтенант получает от генерала. Лейтенант i действует как генерал в алгоритме $OM(m-1)$, чтобы послать значение v_i каждому из $n-2$ других лейтенантов.
3. Для каждого i и каждого $j \neq i$ пусть v_j будет значением, которое лейтенант i получил от лейтенанта j на шаге 2 (с использованием алгоритма (m-1)). Лейтенант i использует большинство значений $(v_1, v_2, \dots, v_n)$

Пример $n = 4, m = 1$:

Подробно покажем пример решения для $n = 4, m = 1$ [1]:

1-й шаг. Каждый генерал посылает всем остальным сообщение, в котором указывает численность своей армии. Лояльные генералы указывают истинное количество, а предатели могут указывать различные числа в разных сообщениях. Генерал 1 указал число 1 (одна тысяча воинов), генерал 2 указал число 2, генерал 3 (предатель) указал трём остальным генералам соответственно x, y, z (истинное значение – 3), а генерал 4 указал 4.

2-й шаг. Каждый формирует свой вектор из имеющейся информации:

Вектор генерала №1: $(1,2,x,4)$;

Вектор генерала №2: $(1,2,y,4)$;

Вектор генерала №3: $(1,2,3,4)$;

Вектор генерала №4: $(1,2,z,4)$.

3-й шаг. Каждый посылает свой вектор всем остальным (генерал 3 посылает опять произвольные значения).

После этого у каждого генерала есть по четыре вектора:

g1	g2	g3	g4
(1,2,x,4)	(1,2,x,4)	(1,2,x,4)	(1,2,x,4)
(1,2,y,4)	(1,2,y,4)	(1,2,y,4)	(1,2,y,4)
(a,b,c,d)	(e,f,g,h)	(1,2,3,4)	(i,j,k,l)
(1,2,z,4)	(1,2,z,4)	(1,2,z,4)	(1,2,z,4)

4-й шаг. Каждый генерал определяет для себя размер каждой армии. Чтобы определить размер i-й армии, каждый генерал берёт (n-m) чисел — размеры этой армии, пришедшие от всех командиров, кроме командира i-й армии. Если какое-то значение повторяется среди этих $(n - m)$ чисел как минимум $(n - m - 1)$ раз, то оно помещается в результирующий вектор, иначе соответствующий элемент результирующего вектора помечается неизвестным (или нулём и т. п.).

Все лояльные генералы получают один вектор $(1,2,f(x,y,z),4)$, где $f(x,y,z)$ есть число, которое встречается как минимум два раза среди значений (x,y,z), или «неизвестность», если все три числа (x,y,z) различны. Поскольку значения x,y,z и функция f у всех лояльных генералов одни и те же, то согласие достигнуто.

Подробности реализации

Канальный уровень:

Нам необходимо, чтобы все сообщения между генералами (лейтенантами) точно были доставлены. Также нам важен порядок доставки сообщений. Потому для реализации канала связи между генералами будем использовать протокол Go-Back-N.

Протокол канального уровня GBN реализован в файле channel_protocol.py

В начале программы каждая пара генералов создаёт линию связи. При отправке сообщения по каналу, в дело вступает отдельный поток, в задачу которого входит организация доставки сообщения (отправка, контроль получения, повторная отправка в случае необходимости). Этот поток «зашит» в канал связи, за счёт чего процесс-отправитель не блокируется на этапе отправки, абстрагируется от реализации протокола и точно уверен в том, что сообщение дойдёт получателя.

Вероятность потери сообщений равно 0.3. Однако протокол всё равно гарантирует, что сообщения будет доставлено.

Сетевой уровень

Алгоритм византийских генералов подразумевает связь каждый с каждым. Так как основной интерес для нас представляет реализация именно протоколов взаимодействия, то организуем сетевой уровень аналогично лабораторной 2 (протокол OSPF).

Сетевой протокол OSPF описан в файле network_protocol.py

Топология роутеров представляется в виде графа. (файл topology.py). Данная структура поддерживает операции добавления/удаления узлов, добавление/удаление связей между узлами. Также в ней содержится реализация поиска кратчайших путей при помощи алгоритма Дейкстры.

В файле network_protocol.py находятся реализации узлов (Router) и выделенного узла (DR – DesignatedRouter). Для подключения к сети роутер сначала устанавливает связь с DR, посылает DR своих соседей, запрашивает у DR текущую топологию сети, после чего приступает к обработке сообщений.

Теперь опишем действия DR при подключении нового узла:

Создать связь.

Получить соседей, после чего необходимо отправить информацию о них все прочим роутерам (кроме отправителя)

Выдать по запросу топологию сети новому узлу

Пример работы программы

Рассмотрим работу алгоритма на примере четырёх генералов, где третий оказался предателем (нумерация будет начинаться с нуля).

В качестве сообщения каждый генерал отправляет свой порядковый номер. Предатель отправляет случайное значение в отрезке [0, n-1]

Сообщения, которые получили генералы:

```
3: [0, 1, 2, 0]
1: [0, 1, 2, 1]
0: [0, 1, 2, 3]
2: [0, 1, 2, 0]
```

Сформированные наборы:

```
0: [[0, 1, 2, 3], [0, 1, 2, 1], [0, 1, 2, 0], [1, 1, 2, 2]]
3: [[0, 1, 2, 3], [0, 1, 2, 1], [0, 1, 2, 0], [2, 2, 1, 2]]
1: [[0, 1, 2, 3], [0, 1, 2, 1], [0, 1, 2, 0], [2, 1, 1, 1]]
2: [[0, 1, 2, 3], [0, 1, 2, 1], [0, 1, 2, 0], [2, 2, 1, 2]]
```

Результаты:

```
0: [0, 1, 2, None]
1: [0, 1, 2, 1]
2: [0, 1, 2, None]
3: [0, 1, 2, None]
```

Как мы видим все лояльные генералы получили идентичную информацию о значениях друг друга. Следовательно, согласие достигнуто. Как мы видим, лишь первый генерал смог сделать выводы о значении предателя, так как для него предатель случайно повторил значение 1

Заключение

Была реализована программа на языке Python для моделирования взаимодействия между генералами (независимыми узлами) на сетевом и канальном уровне. Также программа содержит решение задачи византийских генералов (моделирование наличия злоумышленника в сети).

Показана работоспособность алгоритма на примере $n = 4$ генералов и $m = 1$ злоумышленников.

Приложение:

Ссылка на проект с кодом реализации:
https://github.com/MChepulis/CompNetworks-Labs/tree/main/Lab_2

Использованная литература

1. Wikipedia, Задача византийских генералов, электронный ресурс https://ru.wikipedia.org/wiki/Задача_византийских_генералов
2. The Byzantine Generals Problem https://marknelson.us/posts/2007/07/23/byzantine.html
3. А.Н. Баженов, Компьютерные сети, курс лекций
4. Чернова В.С. Курсовая работа по дисциплине «Компьютерные сети»