Token ring

Требуется реализовать разновидность протокола Token ring.

Описание Token ring

N узлов в сети логически организованы в кольцевую топологию. В сети по кольцу в одном нарпавлении перемещается token, когда в сети нет активных передач то token пуст.

Если узел i хочет передать информацию узлу j, то узел i ожидает пока до него дойдёт пустой token, добавляет в token адрес назначения j и данные для передачи и отправляет узел дальше по кольцу.

Узлы между i и j передают token дальше, не изменяя его. Узел j при получении token вычитывает из него данные и вместе с token отправляет уведомление i о том, что данные успешно доставлены. После того, как узел i получает уведомление об успешной доставке, он "освобождает"токен и передает дальше по сети пустой токен.

Каждый узел удерживает токен в течении времени t.

Задание

Требуется реализовать разновидность протокола Token Ring.

Протокол должен корректно работать в следующих сценариях:

1. Потеря token (пустого и с данными)
2. Выход из строя одного из N узлов
3. Восстановление узла в сети

При выходе из строя узла k предыдущий узел должно начать передавать token следующему после k узлу. При восстановлении узла k передача token должна идти так же, как в изначальной сети (до сбоев).

Общение между узлами ведётся посредством UDP.

Каждый узел сети нужно запустить в отдельной goroutine внутри одного процесса. Узлы сети нужно запускать на localhost, номер порта зависит от номера узла в сети и вычисляется по формуле $port = 30000+i$, где i – номер узла.

Помимо порта *port* узлы должны иметь специальный *maintenance port* $= 40000 + i$, где i – номер узла.

Обычные сообщения (не управляющие) передаются в JSON, формат произвольный.

Управляющие сообщения присылаются на *maintenance port* в формате json.

Логи

При получении token узел должен написать в stdout однои из следующих сообщений:

```
node 2: recieved token from node 1, sending token to node 3

node 2: recieved token from node 1 with data from node 5\
(data='bla-bla-bla'), sending token to node 3

node 5: recieved token from node 4 with delivery confirmation\
from node 2, sending token to node 1
```

При получении управляющего сообщения узел должен написать в stdout сообщение следующего вида:

```
node 2: recieved service message:\
{"type":"send","dst":23,"data":"bla-bla-bla"}
```

Параметры и запуск

Программа должна принимать на вход два параметра:

1. `--n` – количество узлов в сети
2. `--t` – инетрвал времени в мс, в течении которого узел удерживает токен

   Пример запуска:

   ```
   ./run --n 20 --t 10
   ```

Формат управляющих сообщений

Типы сообщений:

1. send – узел на который поступило это сообщение должен отправить данные data узлу, еказанному в dst.

2. terminate – после получения этого сообщения узел перестает слушать на порту *port*

3. recover – после получения этого сообщения узел заново начинает слушать на порту *port*

4. drop – узел должен "потерять"токен, полученный после этого сообщения

Сообщения отправляются в формате JSON. Сообщения состоят из 3 полей:

1. type – строка, тип сообщения (send, terminate, recover, drop)

2. dst – номер узла, на который нужно доставить сообщение (существено только для сообщений типа send)

3. data – строка, содрежащая данные (существено только для сообщений типа send)

Пример управляющего сообщения:

```
{
"type": "send",
"dst": 23,
"data": "bla-bla-bla"
}
```

Пример udp-сервера и клиента можно посмотреть по ссылке: `https://goo.gl/ds6wFK`.

Для сдачи задания нужно прислать письмо на `max.shegai@gmail.com` со ссылкой на репозиторий с кодом. В репозитории помимо кода должен содержаться построенный график.

Репозиторий может быть на любом трэкере (github, gitlab, bitbucket итд).

В теме письма ОБЯЗАТЕЛЬНО указать номер группы.