SERVICIO STREAMING

getTweets(string mensaje) lee 25 tweets y los almacena en mensaje con el formato "$0tweet$1tweet$...$24tweet"

channel of string operation

Process streaming

string pedido operation => (pedido)

while(pedido != "FIN")

if (pedido = "GET_TWEETS")

string mensaje

getTweets(mensaje)

tweets <= (mensaje)

end if

operation => (pedido)

end while

end process

El servicio de *streaming* procesa el fichero de tweets eliminando los caracteres extraños y aquellos tweets que no contengas los campos necesarios y los envía al proceso *master* hasta que reciba un mensaje de fin de parte de este.

MASTER-WORKER

channel of (string) [1..N_WORKERS]worker_gestor

channel of (string) master_gestor, master_streaming

Process worker i := (1..N_WORKERS)

string recibido, datosTags, datosQoS

while(!finalizado)

worker_gestor(i) <- “READ_TAREAS, i”

worker_gestor(i) -> recibido

if(recibido = “FIN”)

finalizado := true

else

datosTags := procesar_tags()

datosQoS := procesar_QoS()

worker_gestor(i) <- “PUBLISH_TAGS, datosTags”

worker_gestor(i) <- “READ_QOS, datosQoS”

end if

end while

end process

Process master

string bloqueTweets, tweets

for i := 0..N_VECES

master_streaming <- “GET_TWEETS”

master_streaming -> bloqueTweets

tweets := procesar_tweets()

master_gestor <- tweets

end for

master_streaming <- “FIN”

master_gestor <- “FIN”

end process

Este modelo conta de dos tipos de proceso: el proceso *master* que se encarga de enviar las tareas y un conjunto de procesos *workers* que recibirán las tareas y se encargarán de llevarlas a cabo.

El proceso *master* se encarga de pedir los tweets a procesar y los recibe directamente del servicio de *streaming*, luego los envía al gestor de colas que los almacenará en la cola de tareas hasta que los *workers* empiecen a procesarlos. Como resultado de este procesamiento de mensajes, los *workers* enviarán la información obtenida sobre la calidad de servicio y de los tags al gestor de colas y este los almacenará en sus colas de QoS y de Tags respectivamente.

Además, será el proceso *master* el que se encargará de cerrar la comunicación. Se enviará un mensaje de fin al servicio de streaming y otro al gestor de colas, después este se encargará de informas a los *workers* y a los analizadores del cierre de la comunicación.

Los procesos *workers* estarán continuamente pidiendo nuevos tweets a procesar al gestor de colas y enviarán la información obtenida hasta que reciban el mensaje de fin del gestor de colas. Como resultado del procesamiento enviarán al gestor de colas información sobre el rendimiento de los propios procesos *worker* e información sobre los tweets que servirá a los procesos analizadores.

GESTOR DE COLAS

El gestor de colas proporcionará las herramientas necesarias para almacenar y gestionar todos los datos del resto de procesos.

Implementación: Primero, es necesario almacenar los datos que el resto de procesos envían. Para ello, se utiliza la cola genérica BoundedQueue ya dada.

Para evitar el solapamiento de entrada y salida de los datos en las colas, se usa un monitor cuyo esquema en alto nivel es el siguiente:

```
Monitor ControldeCola
        integer numElementos := 0
        boolean fin := false
        condition estaEscribiendo
        condition estaBorrando

        operation escribirCola( BoundedQueue cola, string elemento )
                while (numElementos >= MAX_ELEMENTOS)
                        waitC( estaEscribiendo )
                end while
                cola.encolar(elemento)
                numElementos := numElementos + 1
                signalC (estaBorrando)
        end operation

        operation leerCola(BoundedQueue cola, string elemento)
                boolean leido = false
                while (numElementos <= 0 && !fin)
                        WaitC(estaBorrando)
                end while
                if (numElementos > 0)
                        numElementos - -
                        cola.desencolar(elemento)
                        signalC(estaEscribiendo)
```

```
                leido= true;
            end if
            return leido
        end operation

        operation finalizar()
            fin := true
            signalC_all( estaBorrando )
        end operation
end monitor
```

El monitor está diseñado de tal forma que para cada cola que se necesite, se invoque un monitor.

El monitor cuenta con dos variables condición y dos operaciones. Si se quiere escribir en la cola primero se comprueba que la cola no esté llena y si no lo está se añade el elemento deseado al final de la cola. Para leer un elemento de la cola se comprueba primero que la cola no es vacía y si no lo es se guarda el primer elemento y después se desencola. Si fuera vacía, se devuelve falso en un booleano de control.

El programa principal crea dos sockets: uno para el master y los workers y otro para los analizadores. A continuación lanza N + 1 procesos master/worker (siendo N el número máximo de procesos que se pueden conectar) y dos procesos analizadores. A continuación se detalla el funcionamiento de ambos tipos de proceso:

```
channel of string socTareas, socAnalizadores
ControldeCola controlTareas, controlTags, controlQoS
BoundedQueue colaTareas, colaTags, colaQoS

Process masterWorker
        string mensaje
        bool out := false
        while (!out)
            socTareas => mensaje
            if (datos = “FIN”)
                out := true
            else if (mensaje = “PUBLISH_TAREAS,datos”)
```

```
                controlTareas.escribirCola(colaTareas, datos)
        else if (mensaje = “READ_TAREAS”)
                if (controlTareas.leerCola(colaTareas, datos) )
                        socTareas <= datos else mensaje := “FIN” socTareas <= mensaje
                end if
        else if (mensaje = “PUBLISH_QoS”)
                controlQoS.escribirCola(colaQoS, datos)
        else if (mensaje == “PUBLISH_TAGS”)
                controlTags.escribirCola(colaTags, datos)
        end if
    end while
end process
```

El proceso masterWorker es el encargado de gestionar la comunicación con los procesos master/worker dependiendo de lo que escuche por su canal. Por ejemplo si un proceso worker quiere leer tareas pendientes, se desencolara de la cola de tareas los datos correspondientes y se le enviaran estos. Si no quedaran tareas pendientes se le envia un mensaje de fin y se finaliza la comunicación.

```
Process analizadores
    string mensaje
    boolean out := false
    while(!out)
        socAnalizadores => mensaje
        if (mensaje = “READ_QoS”)
            if (controlQoS.leerCola(colaQoS, mensaje))
                socAnalizadores <= mensaje
            else
                mensaje := “FIN”
                socAnalizadores <= mensaje
                out := true
        else if (mensaje =“READ_TAGS”)
            if (controlTags.leerCola(colaTags, mensaje)
                socAnalizadores <= mensaje
```

```
                    else
                        mensaje := “FIN”
                        out := true
                    end if
                end if
            end while
        end process
```

El proceso analizadores se encarga de recibir peticiones de lectura por parte de los analizadores y enviar los datos correspondientes. Por ejemplo: si un analizador quiere leer un dato de la cola de QoS, enviará a través del canal socAnalizadores el mensaje “READ_QoS” y el gestor de colas desencolará el primer elemento que haya en la cola de QoS y se lo enviará al analizador.

ANALIZADORES

1. Analizador de tags

El analizador de Tags proporcionará información de los tweets desde diferentes perspectivas.

- Visión global:

  . Top 5 Clientes más utilizados para acceder a twitter

  · Top 5 Clientes más utilizados para escribir tweets

  · Top 5 Clientes más utilizados al escribir tweets con tags

  · Top 5 Clientes más utilizados al escribir tweets con menciones

  · Top 5 Clientes más utilizados para hacer retweet

  · Porcentaje de ocupación del cliente

- Visión desde los tags:

  · Top 10 Hashtags más utilizados durante la vida del sistema

  · Porcentaje de apariciones del Hashtag del número total de tweets

  · Porcentaje de apariciones del Hashtag del número total de Hashtags

- Visión desde los usuarios:

  · Top 10 usuarios con más actividad

- Visión desde las menciones:

  · Top 10 autores con más menciones

  · Porcentaje de apariciones de la mención del número total de tweets

  · Porcentaje de apariciones de la mención del número total de Hashtags

El analizador de tags se comunica con el gestor de colas y recibe la información a analizar y la trata mediante el siguiente proceso:

```
channel of string socGestor
Process analizadorTags
        string mensaje
        boolean out := false
        while(!out)
                mensaje := “READ_TAGS”
                socGestor <= mensaje
                socGestor => mensaje
                if(mensaje = “FIN”)
                        out := true
                else
                        procesar() // trocea el mensaje recibido, produce los resultados de la
                                // información respecto a cada cliente (información numérica)
                                // guardándolo en variables.
                        almacenarTxt() // trocea el mensaje dejando los hashtags, los manda a
                                        // su fichero. Lo mismo para el autor y las menciones.
                end if
        end while
        tagsInformation() // cliente que procesa su fichero de texto y muestra la información
                        // relativa a los hashtags

        authorsInformation() //cliente que procesa su fichero de texto y muestra la información
                        // relativa a los autores.

        mencionInformation() // cliente que procesa su fichero de texto y muestra la
                        // información relativa a las menciones.
end process
```

2. Analizador de rendimiento

El analizador de rendimiento pretende mostrar los siguientes resultados:

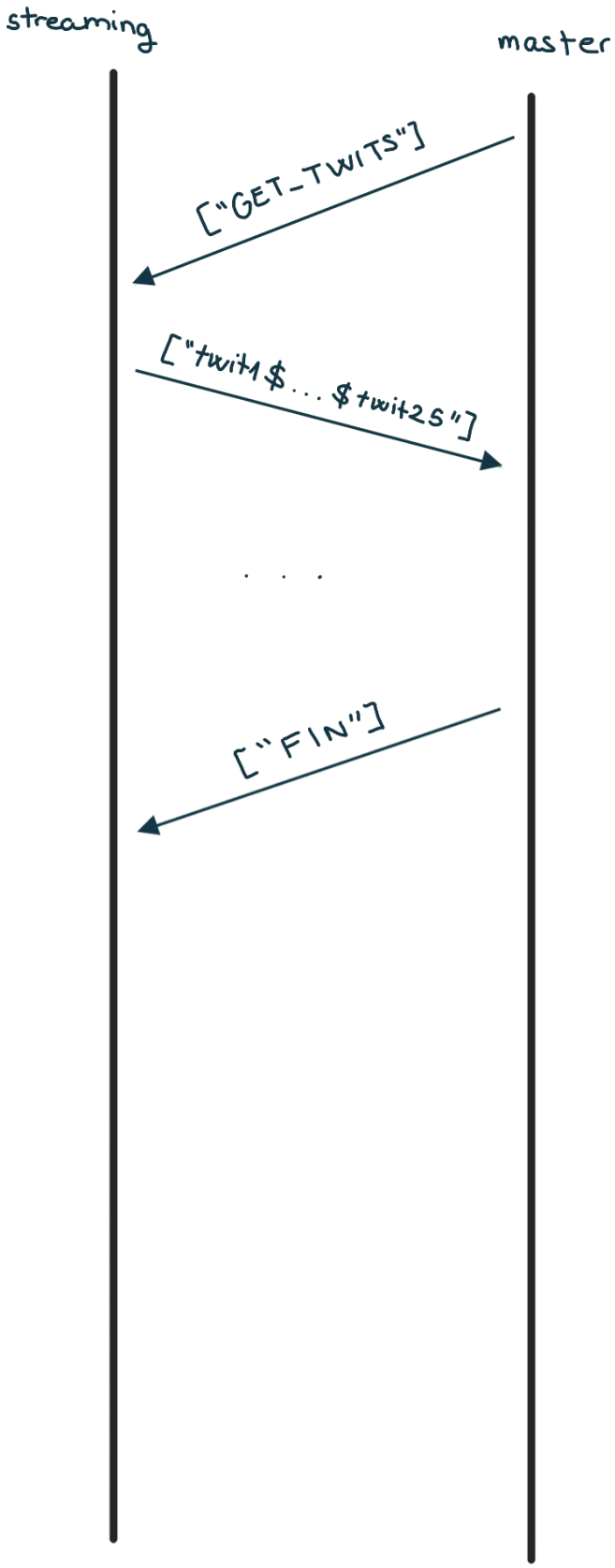
- tiempo medio que le cuesta resolver a cada worker 200 bloques de tareas
- gráfico temporal (para cada worker) tiempo de ejecución cada 200 bloques
- worker más lento y el más rápido (rango en que oscila el tiempo de ejecución medio)
- bloque más lento y más rápido

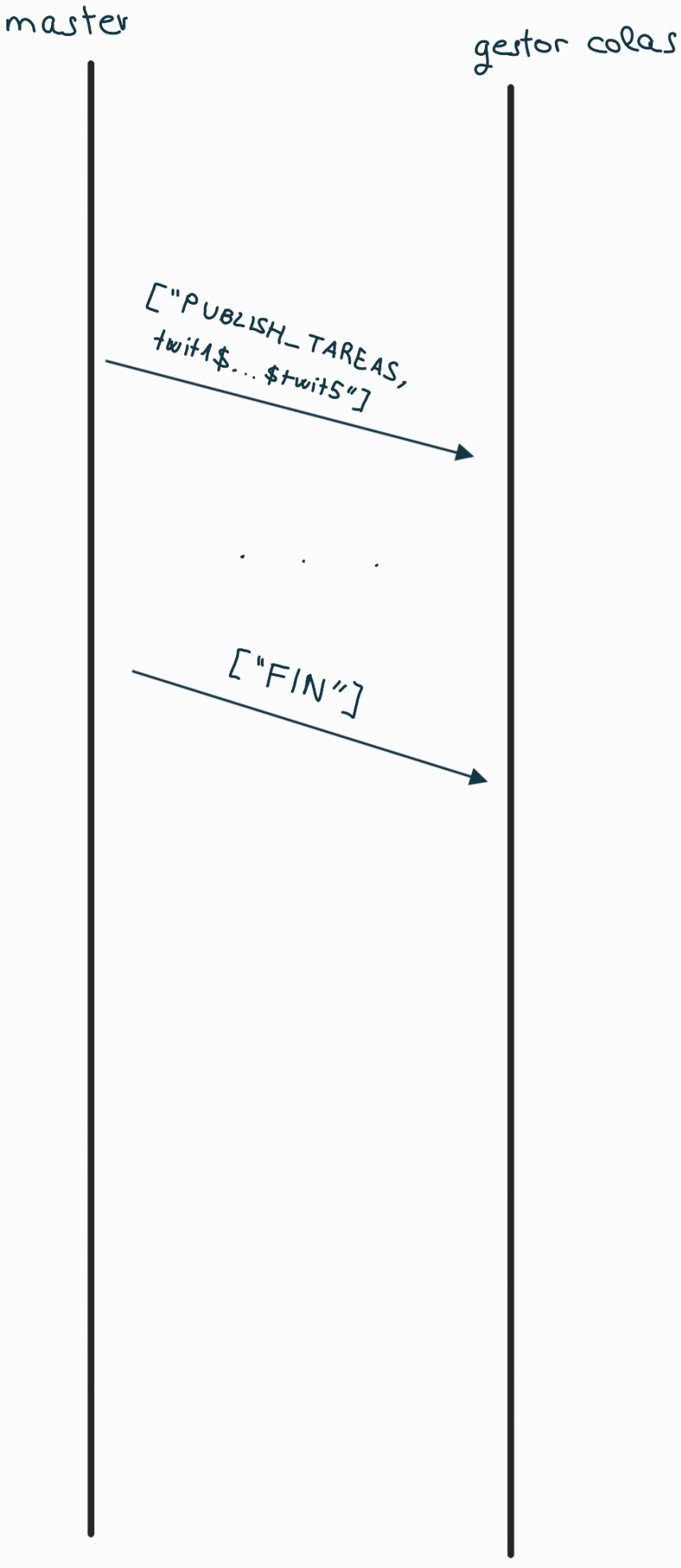
El analizador de rendimiento se comunica con el gestor de colas y recibe la información a analizar y la trata mediante el siguiente proceso:

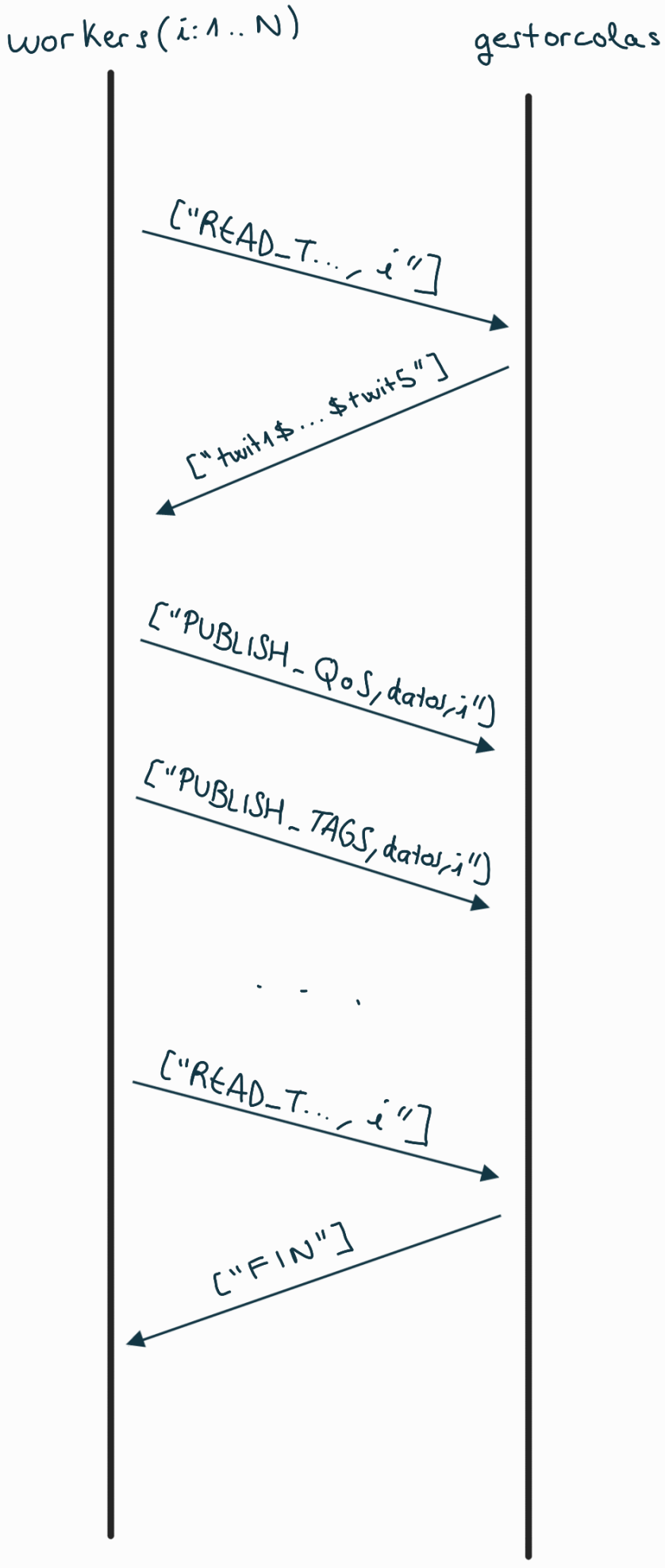
```
channel of string socGestor

Process analizadorQoS

    string mensaje

    boolean out := false

    while(!out)

        mensaje := “READ_QoS”

        socGestor <= mensaje

        socGestor => mensaje

        if(mensaje = “FIN”)

            out := true

        else

            procesarRendimiento()

        end if

    end while

end process
```

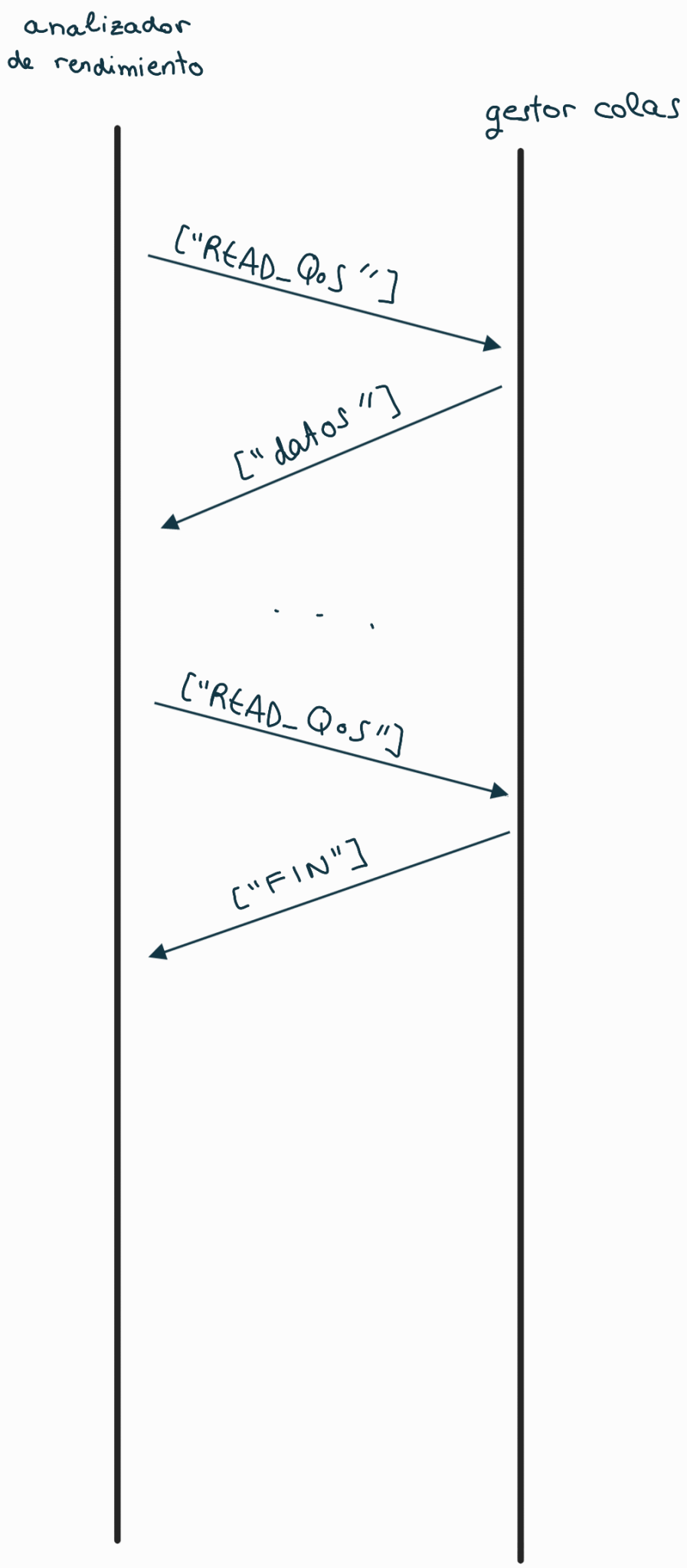
Al cerrar la conexión el analizador de rendimiento mostrará toda la información procesada por pantalla.

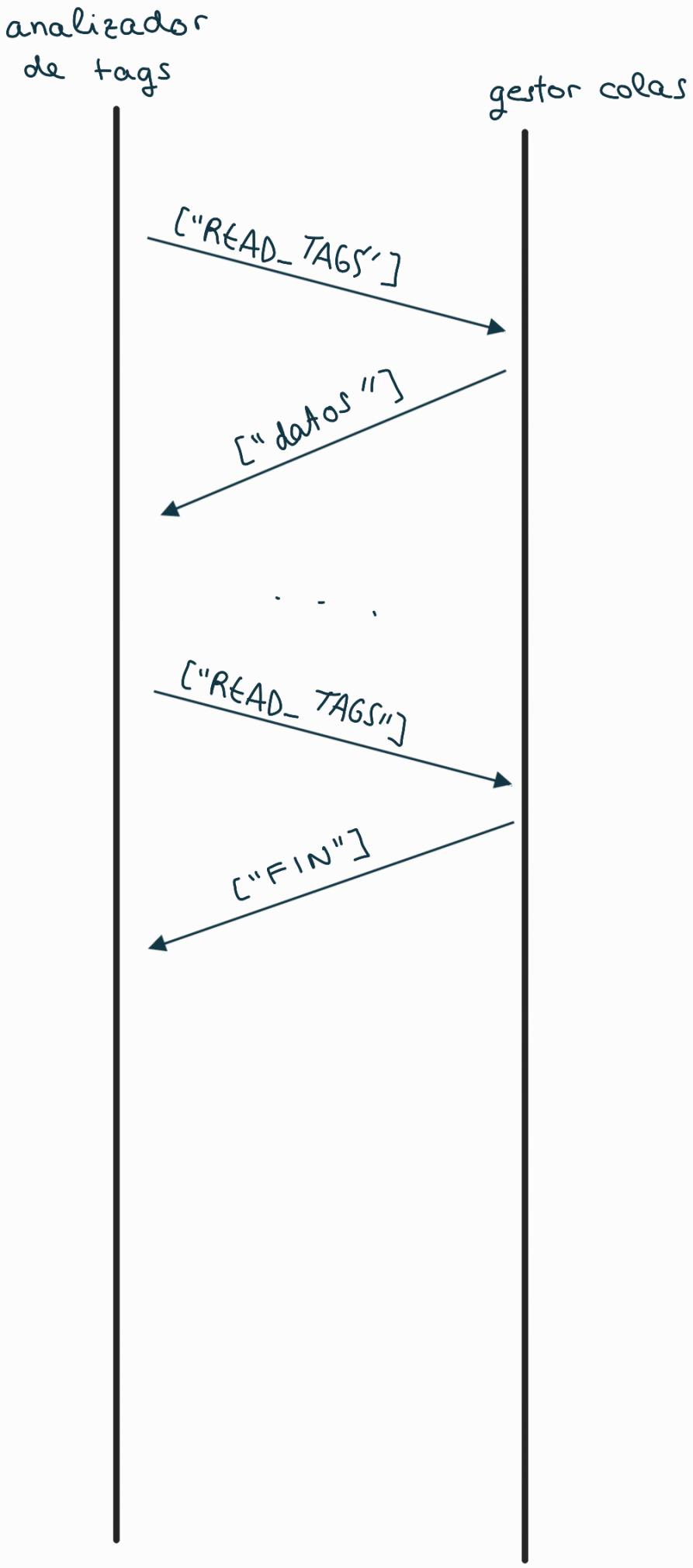
DISEÑO DE COMUNICACIONES




master
gestor colas
["PUBLISH_TAREAS,
twit1$...$twit5"]
. . .
["FIN"]


workers(i:1..N)
gestorcolas
["READ_T..., i"]
["twit1$...$twit5"]
["PUBLISH_QoS, data1, i"]
["PUBLISH_TAGS, data1, i"]
["READ_T..., i"]
["FIN"]


analizador
de rendimiento
gestor colas
["READ_QoS"]
["datos"]
["READ_QoS"]
["FIN"]


analizador de tags
gestor colas
["READ_TAGS"]
["datos"]
. . .
["READ_TAGS"]
["FIN"]