Туториал: мастер-экземляр платформы ("балансировщик" нагрузки)

Балансировка нагрузки Лямбда-Мю 3

Балансировка нагрузки Лямбда-Мю 3 – метод распределения запросов между несколькими экземплярами платформы Лямбда-Мю 3 с целью оптимизации использования ресурсов, резервирования ресурсов, сокращения времени обработки запросов, обеспечения отказоустойчивости и возможности масштабирования такого кластера.

Балансировка нагрузки может решать различные задачи и содержать специфические методы для конкретного проекта, например:

- ассиметричная нагрузка
- коммутация данных из одного формата в другой
- упаковку нескольких запросов от разных клиентов в один пакет запросов
- буферизация данных от медленных клиентов
- фильтрация запросов, не соответствующих протоколу
- проверка работоспособности экземпляров платформы
- аутентификация клиента
- пр.

Таким образом, «балансировщик» нагрузки может взять на себя дополнительные не дорогие в отношении ресурсов операции, которые позволят сосредоточить ресурсы экземпляров платформы на более приоритетных операциях.

Вопрос балансировки нагрузки нужно решать на ранних стадиях проекта. Балансировка непосредственно влияет на корректное, быстрое, отказоустойчивое взаимодействие между клиентом и платформой. Не принятые вовремя меры могут привести к серьёзным последствиям, в том числе материальным.

Пример реализации «балансировщика»

Платформа Лямбда-Мю 3 может использоваться как back end сервер веб-проекта. Одним из таких проектов является «Сомнология B2C» – мобильное приложение на Android. Продемонстрируем реализацию «балансировщика» этого проекта. Основными требованиями являются:

- проверка работоспособности экземпляров платформы
- возврат клиенту данных о наиболее предпочтительном сервере.

Эти требования будут реализованы дополнительным экземпляром платформы Лямбда-Мю 3. Реализация продемонстрирует взаимосвязи нескольких экземпляров платформы Лямбда-Мю 3 с меняющемся количеством логических машин и сторонних клиентов.

Репозитарий

...

Исходные данные

Проект «Сомнология B2C»:

- back end: платформа Лямбда-Мю 3 [server]
- front end: Android-приложение [client]

Нагрузка на [server] от разных [client] одинаковая.

Условия работоспособности [server]:

- к нему можно подключиться (корректно работает объект платформы TCP-server)
- при инициализации пользователя возвращается имя объекта Task (корректно запустилась логическая машина)
- корректный ответ на запрос о количестве клиентов

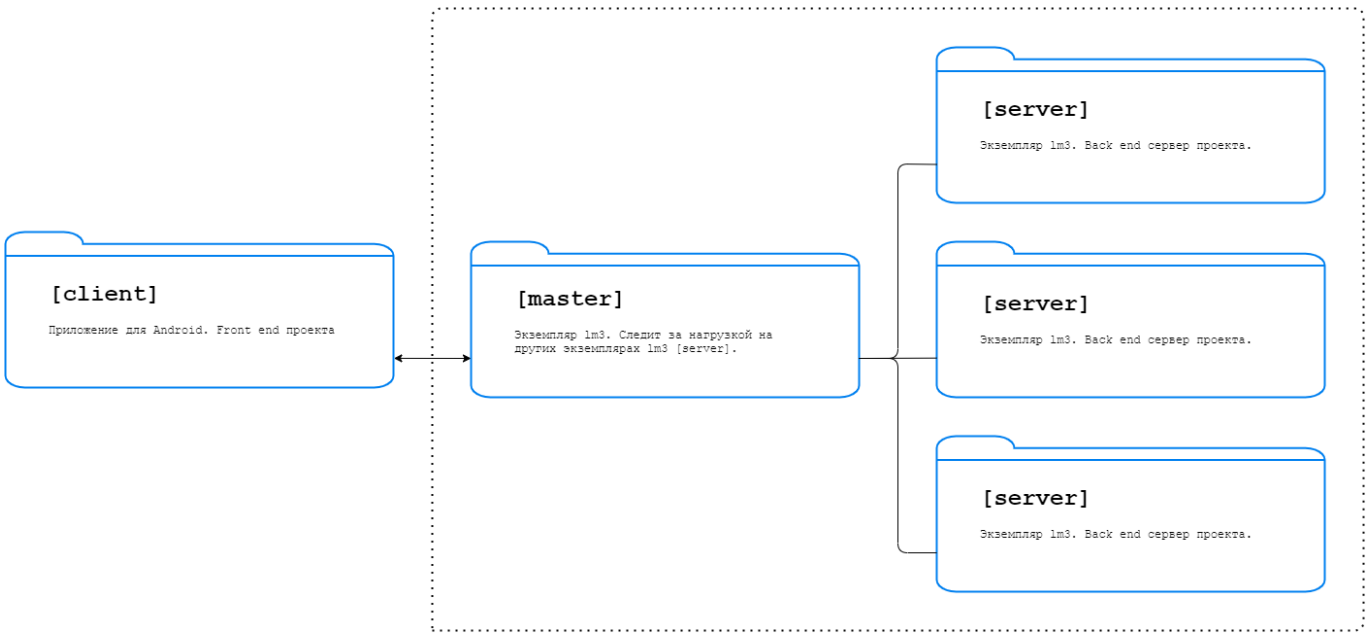
Для отладки решения будут использоваться следующие версии:

- [client] : Android-приложение с вырезанным пользовательским интерфейсом и бизнес-логикой
- [server] : платформа Лямбда-Мю 3 с вырезанной бизнес-логикой

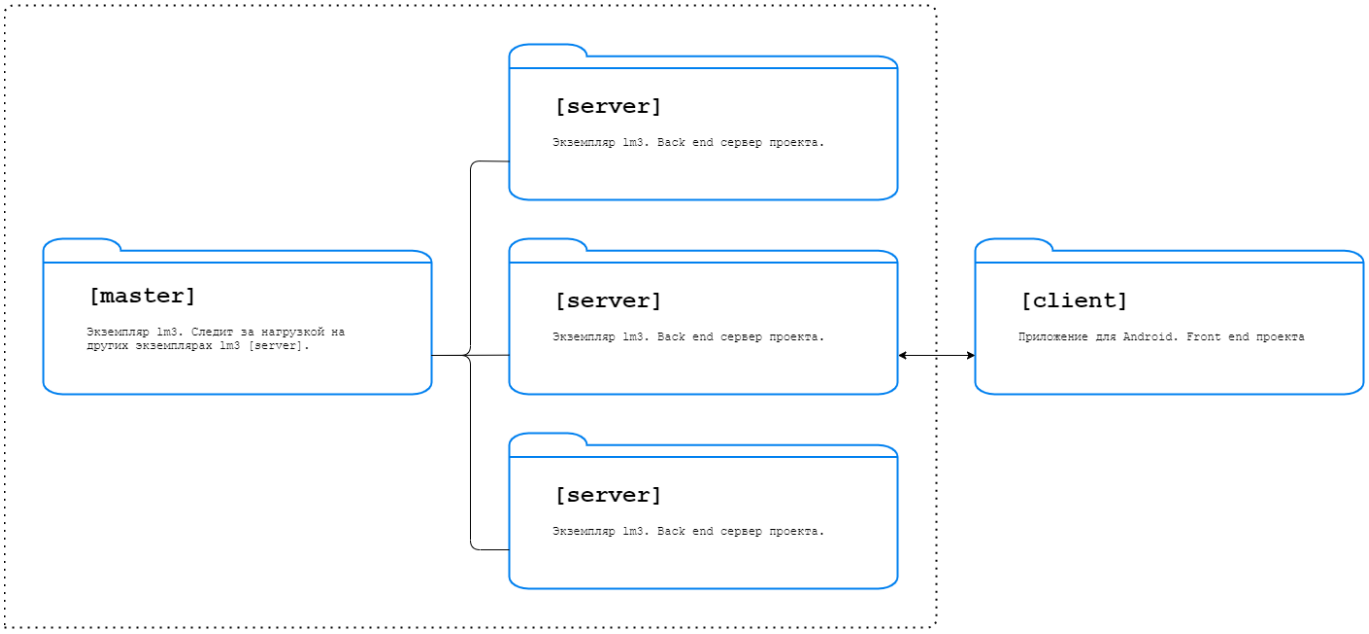
Цель проекта

Распределить нагрузку на экземпляры платформы [server]. При этом клиенты [client] будут сначала выполнять подключение к «балансировщику» [master], получать данные о наиболее подходящем экземпляре [server] и подключаться к нему.

Первый этап:



Второй этап:



Требования

Требованием к проекту является минимальные изменения исходного проекта [client] и [server]. Требование достигается использованием того же протокола, который разработан в исходном проекте.

[client] ↔ [master]

Запрос:

```
{ { Object = "Master", Command = "Event", Name = "DoSendServer" } }
```

Ответ:

```
{ { Object = "Master", Command = "Value", Name = "Server", Data =
<table[Result:string,[Data]:table[IP:string,Port:int]]> } }
```

[master] ↔ [server]

Запрос:

```
{ { Object = "Auth", Command = "Set", Name = "UserName", Data = <string>}, {
Object = "Auth", Command = "Set", Name = "UserMail", Data = <string> } }
```

Ответ:

```
{ { Object = "Auth", Command = "Value", Name = "TaskObject", Data = <string>
} }
```

Запрос:

```
{ { Object = "System", Command = "Get", Name = "UserCount" } }
```

Ответ:

```
{ { Object = "System", Command = "Value", Name = "UserCount", Data = <int> }
}
```

Файловая структура проекта



Взаимосвязи

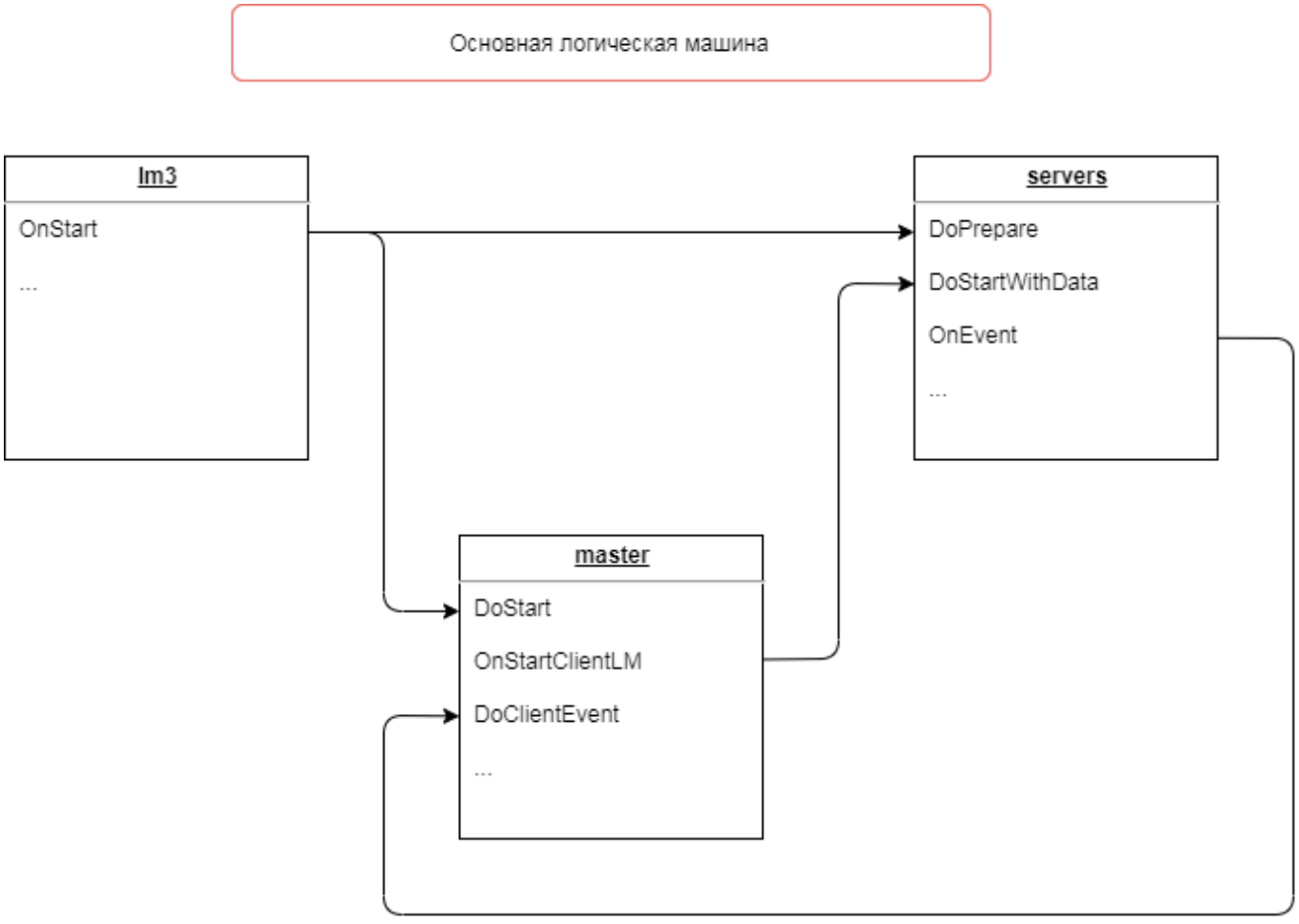
Объекты основной логической машины:

- TCP-server server: прием и передача данных клиенту
- DataPack packer: распаковка и запаковка данных клиенту
- Master master: обработка запросов клиентов и актуализация данных о экземплярах
- Servers servers (наследник LMList): взаимосвязь между ЛМ и объектом master
- Timer timer: по тику логируется существование хотя бы одного работающего экземпляра платформы из пула.

При старте основной логической машины

- вызывается DoPrepare объекта servers - проверяется на валидность конфигурация логических машин
- вызывается DoStart объекта master - цепочка ивентов, создающая для каждого экземпляра платформы (из конфигурационного файла основной логической машины) логическую машину с соответствующими параметрами
- активируется таймер timer

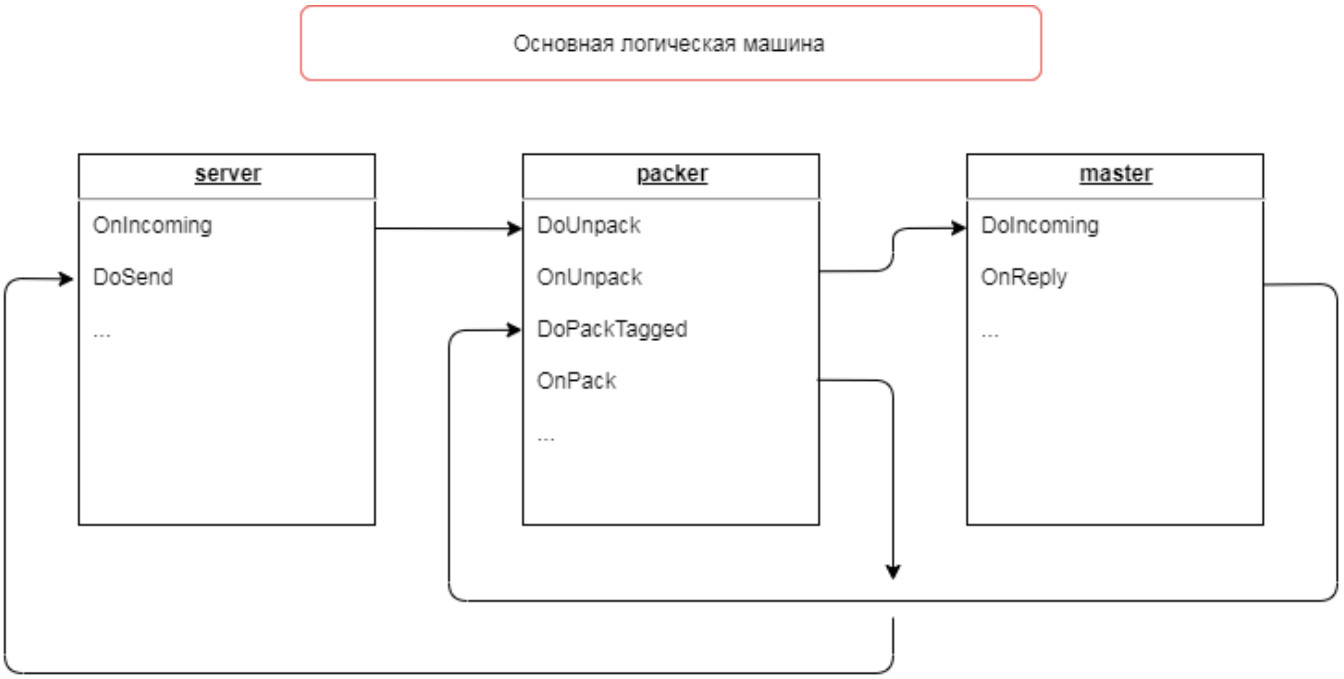
Диаграмма связей:



Взаимодействие с клиентом происходит следующим образом

- server получает запрос
- packer распаковывает данные
- master ищет наиболее подходящий экземляр
- packer пакует данные
- server отправляет ответ

Диаграмма связей:



Объекты логических машин:

- TCP-client client: прием / передача данных экземпляра платформы
- DataPack packer: распаковка / запаковка данных экземпляра платформы
- Timer timer: подключение / отключение к экземпляру платформы
- Timer timerSend: отправка запроса экземпляру платформы

Рассмотрим как устроена проверка на логических машинах.

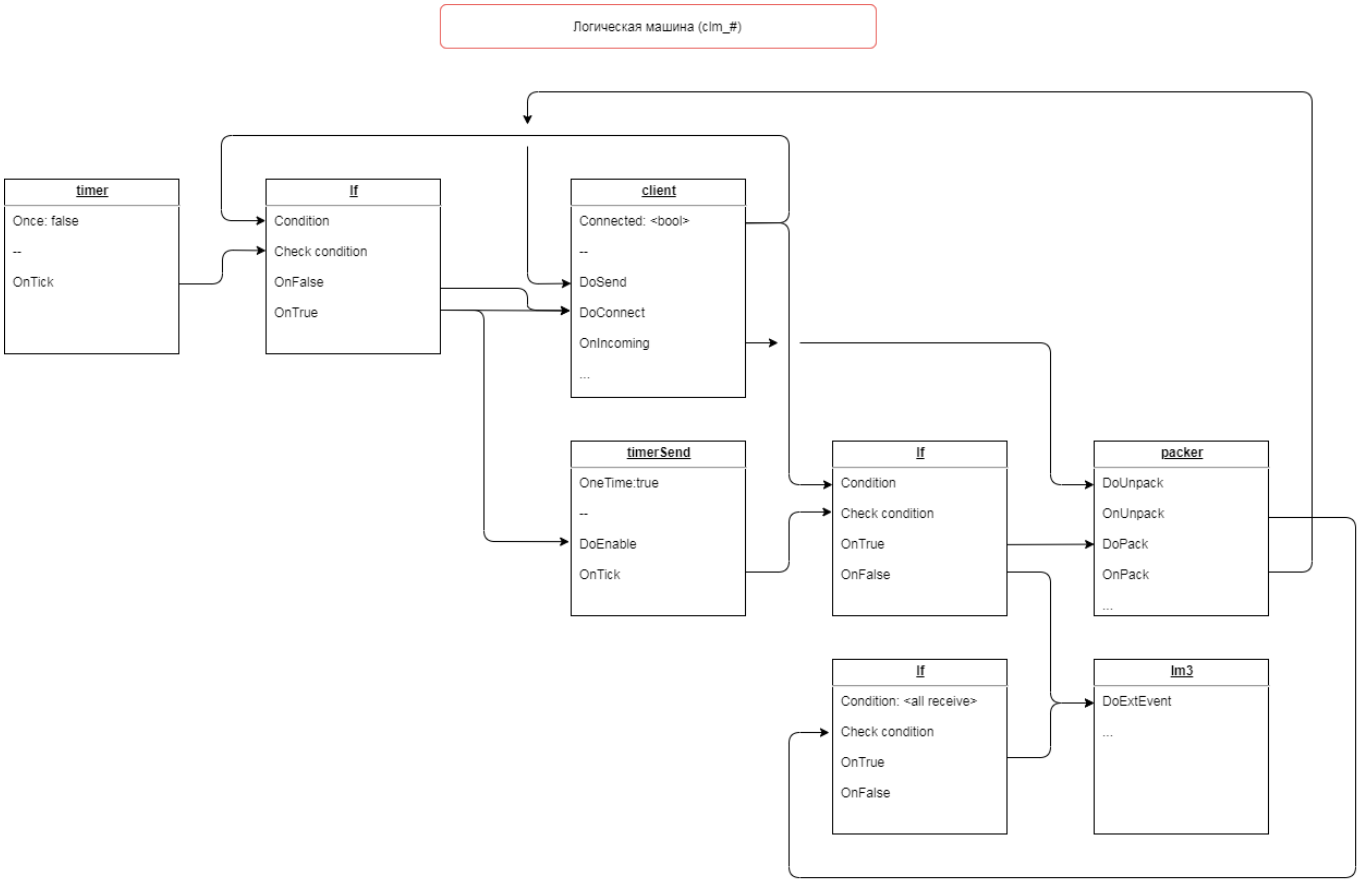
По тику таймера timer происходит подключение / отключение объекта client к проверяемому экземпляру.

Если происходит подключение, то активируется таймер timerSend, по которому при удачном подключении отправится запрос, а при неудачном вернется на основную логическую машину соответствующий результат.

Отправляемый запрос пакуется packer и отправляется client. Ответа ожидается два: количество клиентов и имя объекта Task. Если оба ответа пришли на основную логическую машину возвращается соответствующий результат.

Условия, при которых ничего не отправляется на основную логическую машину, перекрываются отметками времени TimeStamp: если информация старая, то она невалидная.

Диаграмма связей:



Скрипты

main / start.lua

```
lm3:Log("Starting master server... ServersCount =
"..tostring(#lmconf.Servers))

lm3:LoadLibrary( {Alias = "sys", FileName = "lm3system"} )
lm3:Include("main_lib.lua")

-- Объект TCP-сервер: прием/передача данных клиентам
lm3:CreateLObject( { LibAlias = "sys", Type = "TCPServer", Name = "server" }
)
server:SetAttr("SocCount", lmconf.MaxClientCount)
server:SetAttr("Port", lmconf.Port)

-- Объект DataPack: распаковка/запаковка данных клиентов
lm3:CreateLObject( { LibAlias = "sys", Type = "DataPack", Name = "packer" }
)
packer:SetAttr("BufCount", lmconf.MaxClientCount)

-- Объект Servers (наследник объекта LMList): работает с локальными ЛМ (по одному
на сервер)
init_Servers("servers")
-- Объект Master (наследник объекта Base) - основной объект мастер-сервера:
-- запускает ЛМ
-- актуализирует данные с ЛМ: информацию о серверах
```

```
-- выдает наиболее подходящий сервер
init_Master("master")

-- Объект Timer: по тику проверяется, что существует хотя бы один живой сервер
lm3:CreateObject( { Type = "Timer", Name = "timer" } )
timer:SetAttr("Interval", 3 * lmconf.Interval)

-- OnStart: выполняется при старте текущей (основной) ЛМ
lm3:AddHandler("OnStart", function(o, ev)
    -- проверяет конфиг остальных ЛМ на валидность
    servers:DoPrepare(lmconf.BasePath.."servers_lm/config.lua")
    -- старт остальных ЛМ
    master:DoStart()
    -- старт проверок
    timer:DoEnable(true)
end)

timer:AddHandler("OnTick", function(o, ev)
    master:DoCheckAtLeastOneExist()
end)

-- связи объектов:
server:Connect("OnIncoming", "packer", "DoUnpack")
packer:Connect("OnUnpack", "master", "DoIncoming")
packer:Connect("OnPack", "server", "DoSend")
master:Connect("OnReply", "packer", "DoPackTagged")
master:Connect("OnStartClientLM", "servers", "DoStartWithData")
master:Connect("OnStopClientLM", "servers", "DoStop")
master:Connect("OnClientEvent", "servers", "DoEvent")
servers:Connect("OnEvent", "master", "DoClientEvent")
```

main / main_lib.lua

```
-- Объект Servers: работает с локальными ЛМ (по одному на сервер)
function init_Servers(_name)
    local obj = lm3:CreateObject( { Type = "LMList", Name = _name } )

    -- обертка над протоколом DoStart
    -- обновляет конфиг данными IP и Port
    obj:AddInProc("DoStartWithData",
"table[Name:string,Data:table[IP:string,Port:int]]", "bool", function(o, ev)
        lm3:Log("DoStartWithData: Name: "..ev.Name.."; IP: "..ev.Data.IP.";
Port: "..ev.Data.Port)
        local conf = obj:GetAttr("ConfigData")
        conf.IP = ev.Data.IP
        conf.Port = ev.Data.Port
        conf.Interval = lmconf.Interval
        --lm3:Log("config: "..lm3:DataToString(conf))
        return obj:DoStart({ Name = ev.Name, Config = conf }) == true
    end)
end
```

```
-- Объект Master - основной объект мастер-сервера:
-- запускает ЛМ
-- актуализирует данные с ЛМ: информацию о серверах
-- выдает наиболее подходящий сервер
function init_Master(_name)
    local obj = lm3:CreateObject( { Type = "Base", Name = _name } )

    -- структура данных о серверах
    obj.Servers = {}
    for i = 1, #lmconf.Servers do
        obj.Servers[i] = {
            Port = lmconf.Servers[i].Port,
            IP = lmconf.Servers[i].IP,
            Alive = false,
            CurrentUsers = 0,
            TimeStamp = 0,
            LMName = "clm_"..i,
            LMConnected = false
        }
    end

    -- DoStart: старт ЛМ
    obj:AddInEvent("DoStart", "nil", function(o, ev)
        lm3:Log("DoStart")
        for i = 1, #obj.Servers do
            lm3:Log("Server: "..i.."; IP: "..obj.Servers[i].IP.."; Port:
"..obj.Servers[i].Port)
            lm3:Log("Start client LM: "..obj.Servers[i].LMName)
            if obj:OnStartClientLM( { Name = obj.Servers[i].LMName, Data = {
IP = obj.Servers[i].IP, Port = obj.Servers[i].Port} } ) == true then
                lm3:Log("Start client LM complete.")
            else
                lm3:Log("ERROR: Start client LM failed.")
                obj.Servers[i].LMConnected = false
            end
        end
    end)

    -- FindServer: находит наиболее подходящий сервер (nil, если не найден)
    function obj.FindServer()
        if #obj.Servers == 0 then
            return nil
        end

        local server = nil

        for i = 1, #obj.Servers do
            if obj.Servers[i].Alive then
                --lm3:Log("os.time(): "..os.time())
                --lm3:Log("stamp    : "..obj.Servers[i].TimeStamp)
```

```
                if (os.time() - obj.Servers[i].TimeStamp) < (3 *
lmconf.Interval / 1000)  then
                    if server == nil then
                        server = obj.Servers[i]
                    else
                        if server.CurrentUsers > obj.Servers[i].CurrentUsers
then
                            server = obj.Servers[i]
                        end
                    end
                else
                    lm3:Log("ERROR! Server: "..obj.Servers[i].IP..";
"..obj.Servers[i].Port.." has old time stamp!")
                end
            else
                lm3:Log("ERROR! Server: "..obj.Servers[i].IP..";
"..obj.Servers[i].Port.." not alive!")
            end
        end

        lm3:Log("FindServer: result: "..lm3:DataToString(server))
        return server
    end

    -- DoIncoming: обрабатывает данные от клиента
    -- на SendServerInfo возвращает сервер (или "not found")
    obj:AddInEvent("DoIncoming",
"table[Tag:int,Data:array[table[Object:string,Command:string,Name:string,[Da
ta]:any]]]", function(o, ev)
        lm3:Log("Incoming data from client with tag: "..tostring(ev.Tag)..."
: "..lm3:DataToString(ev.Data))
        for i = 1,#ev.Data do
            if ev.Data[i].Object == "Master" then
                if ev.Data[i].Command == "Event" then
                    if ev.Data[i].Name == "DoSendServer" then
                        local server = obj.FindServer()
                        if server == nil then
                            obj:OnReply( { Tag = ev.Tag, Data = { {Object =
"Master", Command = "Value", Name = "Server", Data =
                                {
                                    Result = "not found"
                                }
                            }})
                        else
                            obj:OnReply( { Tag = ev.Tag, Data = { {Object =
"Master", Command = "Value", Name = "Server", Data =
                                {
                                    Result = "found",
                                    Data = { IP = server.IP, Port =
server.Port }
                                }
```

```
                                }}})
                            end
                        end
                    end
                end
            end
        end)

        function obj.TagFromLMName(_name)
            return tonumber(_name:sub(5))
        end

        -- DoClientEvent: обрабатывает данные от ЛМ
        obj:AddInEvent("DoClientEvent",
"table[LMName:string,EventName:string,Data:any]", function(o, ev)
            local i = obj.TagFromLMName(ev.LMName)
            obj.Servers[i].Alive = ev.Data.Alive
            obj.Servers[i].CurrentUsers = ev.Data.CurrentUsers
            obj.Servers[i].TimeStamp = ev.Data.TimeStamp
        end)

        -- DoCheckAtLeastOneExist: проверить, что хотя бы один сервер жив
        obj:AddInEvent("DoCheckAtLeastOneExist", "nil", function(o, ev)
            local table = obj.FindServer()
            if table ~= nil then
                lm3:Log("DoCheckAtLeastOneExist: GOOD! Server found")
            else
                lm3:Log("DoCheckAtLeastOneExist: BAD! Server not found")
            end
        end)

        obj:AddOutEvent("OnReply",
"table[Tag:int,Data:array[table[Object:string,Command:string,Name:string,[Da
ta]:any]]]")

        obj:AddOutProc("OnClientEvent",
"table[LMName:string,EventName:string,Data:any]", "bool")

        obj:AddOutProc("OnStartClientLM",
"table[Name:string,Data:table[IP:string,Port:int]]", "bool")

        obj:AddOutProc("OnStopClientLM", "string", "bool")


        return obj
end
```

main / servers_lm / start.lua

```
lm3:Log("Start!")
lm3:Log("IP: "..lmconf.IP.."; Port: "..lmconf.Port)
```

```
lm3:LoadLibrary( {Alias = "sys", FileName = "lm3system"} )

-- объект TCP-клиент: получает/отправляет данные
lm3:CreateLObject( { LibAlias = "sys", Type = "TCPClient", Name = "client" }
)
client:SetAttr("IP", lmconf.IP)
client:SetAttr("Port", lmconf.Port)

-- объект DataPack: запаковывает/распаковывает данные
lm3:CreateLObject( { LibAlias = "sys", Type = "DataPack", Name = "packer" }
)

-- структура данных: результат запроса серверу
result = {}
result.Alive = false
result.CurrentUsers = 0
result.TimeStamp = os.time()

-- две проверки:
-- получить имя объекта Task
-- получить количество клиентов
checkCounter = 2

-- таймер "подключись/отключись от сервера"
lm3:CreateObject( { Type = "Timer", Name = "timer" } )
timer:SetAttr("Interval", lmconf.Interval)

-- таймер "отправь запрос"
lm3:CreateObject( { Type = "Timer", Name = "timerSend" } )
timerSend:SetAttr("Interval", 1000)
timerSend:SetAttr("OneTime", true)

-- OnStart: старт текущей ЛМ
lm3:AddHandler("OnStart", function(o, ev)
    -- активация таймера "подключись/отключись от сервера"
    timer:DoEnable(true)
end)

-- обработчик таймера "подключись/отключись от сервера"
timer:AddHandler("OnTick", function(o, ev)
    checkCounter = 2
    if client:GetAttr("Connected") then
        lm3:Log("timer: OnTick: do disconnect")
        client:DoConnect(false)
    else
        lm3:Log("timer: OnTick: do connect")
        client:DoConnect(true)
        timerSend:DoEnable(true)
    end
end)
```

```
-- обработчик таймера "отправь запрос"
-- если Connected: отправим запрос
-- если не Connected: вернем основной ЛМ Alive = false
timerSend:AddHandler("OnTick", function(o, ev)
    if client:GetAttr("Connected") then
        lm3:Log("timerSend: OnTick: send request")
        packer:DoPack( {
            { Object = "Auth", Command = "Set", Name = "UserName", Data =
"master"},
            { Object = "Auth", Command = "Set", Name = "UserMail", Data =
"master"},
          } )
        packer:DoPack( { { Object = "System", Command = "Get", Name =
"UserCount" } } )
    else
        lm3:Log("timerSend: OnTick: not connected!")
        result.Alive = false
        result.CurrentUsers = 0
        result.TimeStamp = os.time()
        lm3:DoExtEvent(result)
    end
end)

-- обработка данных от сервера
packer:AddHandler("OnUnpack", function(o, ev)
    local data = ev
    for i = 1,#data do
        if data[i].Object == "System" then
            if data[i].Command == "Value" then
                if data[i].Name == "UserCount" then
                    result.Alive = true
                    result.CurrentUsers = data[i].Data
                    result.TimeStamp = os.time()

                    checkCounter = checkCounter - 1
                end
            end
        elseif data[i].Object == "Auth" then
            if data[i].Command == "Value" then
                if data[i].Name == "TaskObject" then
                    checkCounter = checkCounter - 1
                end
            end
        end
    end

    if checkCounter == 0 then
        lm3:Log("client: checkCounter==0, result:
"..lm3:DataToString(result))
        lm3:DoExtEvent(result)
```

```
        checkCounter = 2
    end
end)

-- связи объектов
packer:Connect("OnPack", "client", "DoSend")
client:Connect("OnIncoming", "packer", "DoUnpack")
```

Логи

[master] LM_main.log

[master] LM_clm_1.log

[master] LM_main.log

[server] LM_main.log

[client]

From:
https://wiki.lambda-mu.com/ - **Lambda-Mu Wiki**

Permanent link:
https://wiki.lambda-mu.com/lm3/ce/t_master?rev=1606315030

Last update: **2020/11/30 14:12**