V-NetLab Group-manager Interface Guide

v0.1

Praveen Krishnamoorthy <kpraveen85@gmail.com>

1. Group-manager Interfaces:

grp-mgr is the central daemon which runs on the gateway machine and interfaces with the following components of the V-Netlab.

- Vnet-mgr
- Resource allocator
- Host-mgr

The interfacing with these components are discussed in detail below. As grp-mgr is the core of the V-NetLab, the same interface would also be listed in the respective component's interface document too.

2. Interfacing with vnet-mgr:

vnet-mgr communicates with the grp-mgr through sockets. The */home/vnetlab/conf/group/groupsConfig.txt* specifies the information about the grp-mgr ie, IP address, port, group ID it caters to, etc. Under the currect setup the vnet-mgr connects to the port **7777** on which the grp-mgr listens for incoming requests.

2.1 vnet-mgr → grp-mgr

2.1.1 Header format:

```
typedef struct __HEADER__ {
   int uid ;            //uid of the user
   int gid ;            //supplementary gid of the user, which represents the team id
   int opid ;           //op code
   int team;
   int console;
} header;
```

2.1.2 Payload format:

```
typedef struct __MGR_ST_SD_SEND_PAYLOAD__ {
   int displayNo;
   char  userName[256];
   char networkName[256];
   char cookie[256];
} mgr_st_sd_send_pl;
```

2.2 grp-mgr → vnet-mgr

2.2.1 Header format:

```
typedef struct __HEADER__ {
   int uid ;            //uid of the user
   int gid ;            //supplementary gid of the user, which represents the team id
   int opid ;           //op code
   int team;
   int console;
} header;
```

2.2.2 Payload format:

```
typedef struct __MGR_ST_SD_GRAB_RECV_PAYLOAD__ {
   int status ;                    // Success or failure
   char data[CHAR_BUF_SIZE];       // Detailed info in-case of failure
} mgr_st_sd_gr_recv_pl;
```

3. Interfacing with Resource-Allocator

grp-mgr interfaces with the resource allocator on the port **9000** (GMGR_RMGR_PORT)

3.1 grp-mgr → resource-allocator:

The grp-mgr sends the following data to the resource allocator.

- Filename containing the user requested network information in Chaco format
- Filename containing the physical host information (on which VMs could be allocated)
- target_load – parameter required for resource allocation (a float value)
- Resource allocator algorithm to be used for allocation
- additional_info - specific to round next algorithm

3.2 resource-allocator → grp-mgr

```
typedef struct allocation_results {
    int result;                    // success(1) or failure (0)
    float targetLoad;
    int vmNum;                     // No of vm_allocation structures that follows this ie, no of VMs allocated
    char  info[128];               // if algorithm is round next fit, this will store the name of the host
    vm_allocation vms[0];
} Allocation;

typedef struct __host_info__ {
    char vmname[64];
    float cpu;
    float mem;
    float bw;
   float completRate;
   char hostname[64];
} vm_allocation;
```

4. Interfacing with host-mgr

There are two interfaces between grp-mgr & host-mgr.

- host-mgr communicates with the grp-mgr while starting
- grp-mgr sends across the user requests to host-mgr and the host-mgr replies back once serviced

4.1 host-mgr startup interface with grp-mgr

The communication is through sockets and and the grp-mgr listens in the port (GMGR_HMGR_PORT = **7778**) The host-mgr while starting up contacts the grp-mgr and if the grp-mgr doesn't reply back the host-mgr makes an exit.

4.1.1 host-mgr → grp-mgr

```
typedef struct __GMGR_HMGR_MSG__ {
   gmgr_hmgr_hdr header;
   char data[GrpMgr_HostMgr::MAX_DATA_LEN] ;
} gmgr_hmgr_msg ;

typedef struct __GMGR_HMGR_HEADER__ {
   int opid;      // GrpMgr_HostMgr::JOIN_HOST  or  GrpMgr_HostMgr::DETACH_HOST
   int uid;
   int nid;
   int tid;
   int team;
   int console ;
} gmgr_hmgr_hdr ;
```

4.1.2 grp-mgr → host-mgr

```
typedef struct __GMGR_HMGR_MSG__ {
   gmgr_hmgr_hdr header;
   char data[GrpMgr_HostMgr::MAX_DATA_LEN] ;
} gmgr_hmgr_msg ;

typedef struct __GMGR_HMGR_HEADER__ {
   int opid;   // GrpMgr_HostMgr::SUCCESS   or  GrpMgr_HostMgr::FAILURE
   int uid;
   int nid;
   int tid;
   int team;
   int console ;
} gmgr_hmgr_hdr ;
```

4.2 grp-mgr user request servicing interface with host-mgr

The communication is through sockets and and the host-mgr listens in the port (HMGR_GMGR_PORT = **7798**) grp-mgr sends across the user requests to host-mgr and the host-mgr replies back once serviced

4.2.1 grp-mgr → host-mgr

```
typedef struct __GMGR_HMGR_MSG__ {
   gmgr_hmgr_hdr header;
   char data[GrpMgr_HostMgr::MAX_DATA_LEN] ;
} gmgr_hmgr_msg ;

typedef struct __GMGR_HMGR_HEADER__ {
   int opid;   //GrpMgr_HostMgr::START .... GrpMgr_HostMgr::SHUTDOWN
   int uid;
   int nid;
   int tid;
   int team;
```

```
    int console ;
} gmgr_hmgr_hdr ;
```

4.2.2 host-mgr → grp-mgr

```
typedef struct __GMGR_HMGR_MSG__ {
    gmgr_hmgr_hdr header;
    char data[GrpMgr_HostMgr::MAX_DATA_LEN] ;
} gmgr_hmgr_msg ;

typedef struct __GMGR_HMGR_HEADER__ {
    int opid;   // GrpMgr_HostMgr::SUCCESS   or  GrpMgr_HostMgr::FAILURE
    int uid;
    int nid;
    int tid;
    int team;
    int console ;
} gmgr_hmgr_hdr ;
```

Related Documents:

Please refer to the following documents for other components interface.

- Kernel Module Interface <Manish Mehra>
- Resource Allocator Interface <Mingwei Zhang>
- Host Manager Interface <Rui Qiao>