SSG (Scalable Service Groups)

Mochi Bootcamp
September 24-26, 2019

Session information

Instructions for the SSG tutorial and a copy of these slides can be found by following the “Session 3: SSG” link in the Mochi bootcamp repository README.md

Refer back to “Session 2: Hands-on” for general details on logging onto JLSE, installing Mochi software, and running jobs

Following the tutorial we will install SSG and attempt to run and modify an example distributed service

https://xgitlab.cels.anl.gov/sds/mochi-boot-camp/

Group membership background

Motivation:

Distributed systems frequently require a group membership service to reach agreement on the set of processes comprising the system, even in the face of process failures and growing/shrinking resource allocations

Challenges:

- How do processes learn about the initial membership of a group (i.e., bootstrapping)?
- How do processes distinguish between failed group members and members that are temporarily unresponsive?
- How do processes agree on group membership changes in a consistent manner?

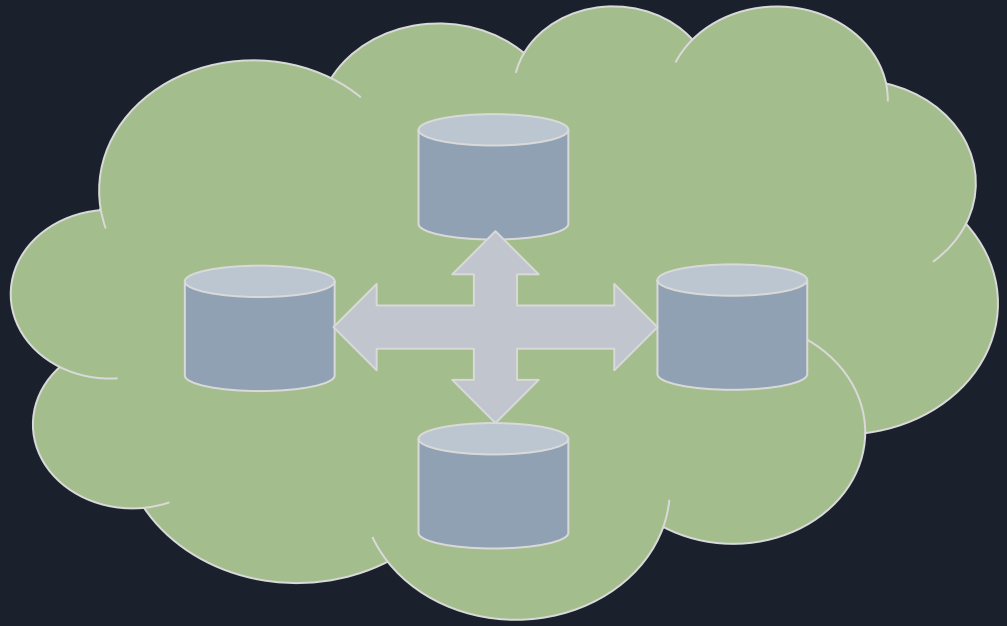
https://xgitlab.cels.anl.gov/sds/mochi-boot-camp/

Motivating group membership use case

Distributed Object Store

Group of servers need to maintain agreement on active membership list to effectively distribute objects

Connections across the group need to be managed scalably and reliably

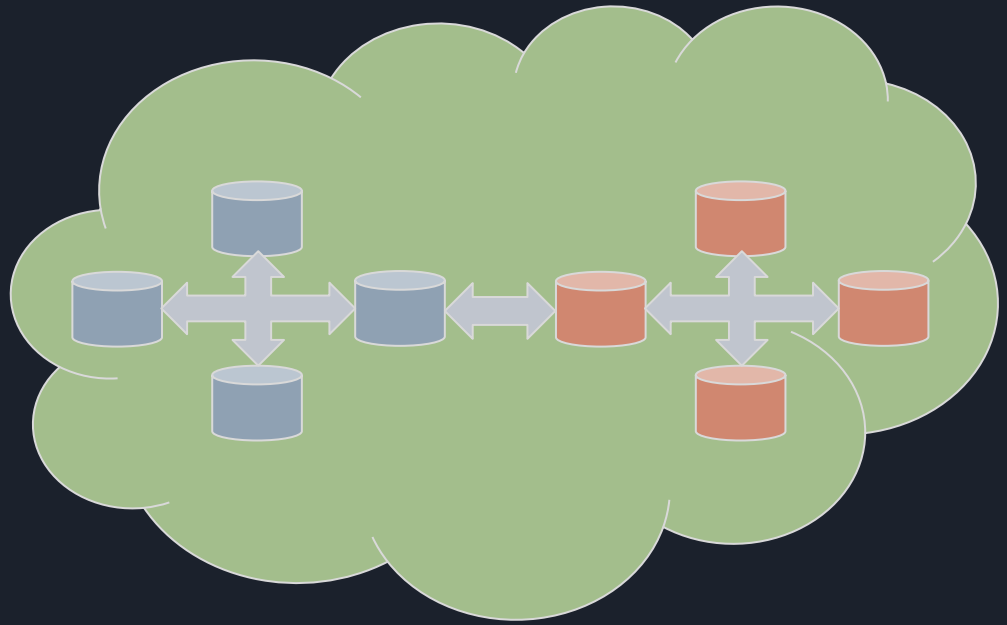


https://xgitlab.cels.anl.gov/sds/mochi-boot-camp/

Motivating group membership use case

Distributed Object Store

Servers may even want to arrange in subgroups, similar to how Ceph organizes into placement groups



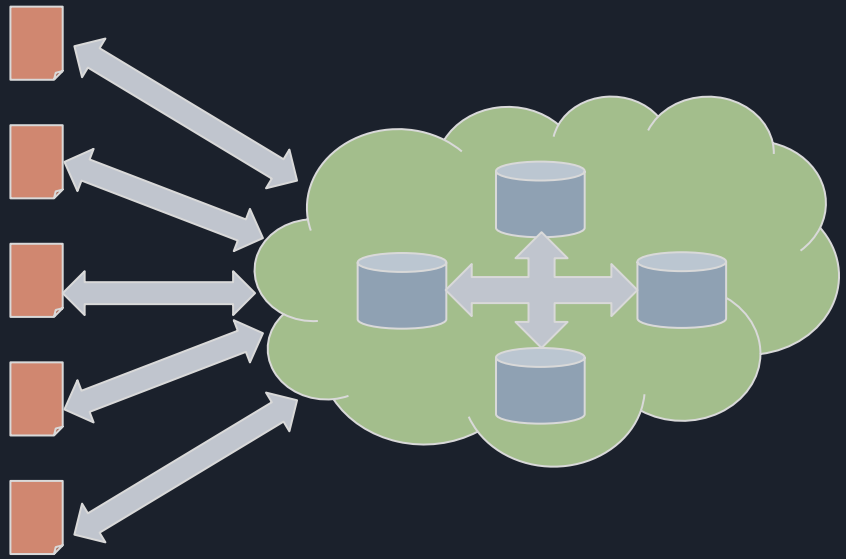
https://xgitlab.cels.anl.gov/sds/mochi-boot-camp/

Motivating group membership use case

Distributed Object Store

Object store clients may also want to “observe” the server group view, so client requests can be load-balanced

Clients likely only want access to group membership snapshot at time of request, not to become active members

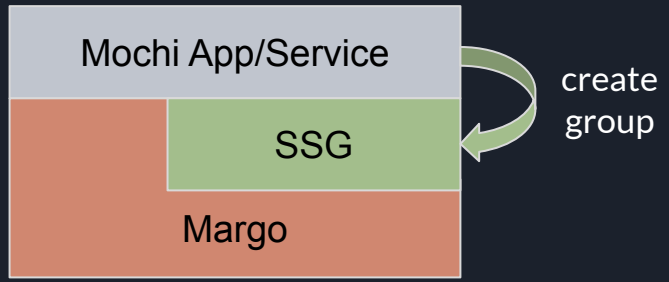


https://xgitlab.cels.anl.gov/sds/mochi-boot-camp/

SSG: A Mochi-based group membership service

SSG is a dynamic group membership service built directly atop Margo that performs the following tasks:

- Bootstraps groups using a number of methods
  - MPI
  - PMIx
  - config file
- Generates unique process IDs for group members and provides member ID -> address mappings (*views*)
- Manages group membership dynamically as processes explicitly join/leave groups or implicitly fail

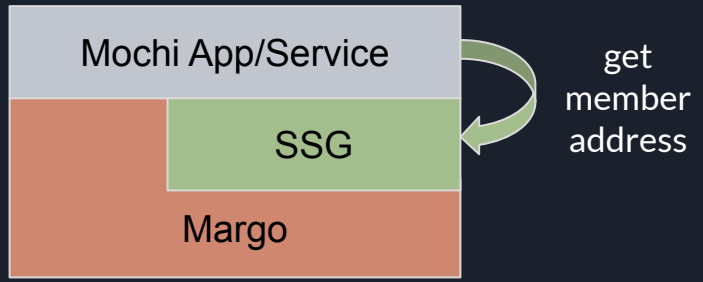


https://xgitlab.cels.anl.gov/sds/mochi-boot-camp/

SSG: A Mochi-based group membership service

SSG is a dynamic group membership service built directly atop Margo that performs the following tasks:

- Bootstraps groups using a number of methods
  - MPI
  - PMIx
  - config file
- Generates unique process IDs for group members and provides member ID -> address mappings (*views*)
- Manages group membership dynamically as processes explicitly join/leave groups or implicitly fail

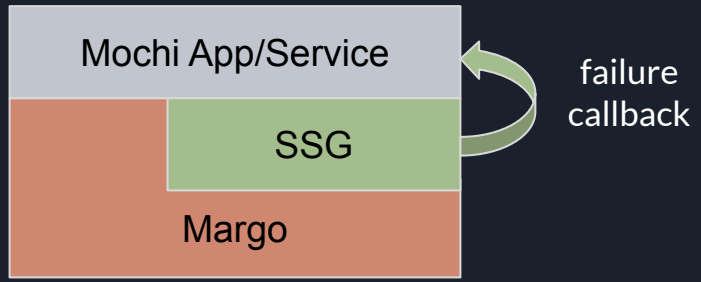


https://xgitlab.cels.anl.gov/sds/mochi-boot-camp/

SSG: A Mochi-based group membership service

SSG is a dynamic group membership service built directly atop Margo that performs the following tasks:

- Bootstraps groups using a number of methods
  - MPI
  - PMIx
  - config file
- Generates unique process IDs for group members and provides member ID -> address mappings (*views*)
- Manages group membership dynamically as processes explicitly join/leave groups or implicitly fail



https://xgitlab.cels.anl.gov/sds/mochi-boot-camp/

SSG is not MPI

SSG is similar to MPI in that it bootstraps communication across a set of processes and uniquely identifies each group member, but does not try to emulate MPI beyond that

- SSG does not provide any sort of collective communication algorithms across a group, just a list of member IDs
  - No broadcast, barrier, reductions, datatype support etc.
- SSG does not even provide wrappers for sending RPCs to group members and instead just provides mappings of member IDs to Mercury addresses

Ultimately, the implementation of collective communication algorithms is left to an additional layer, with SSG focusing solely on membership and fault-tolerance

https://xgitlab.cels.anl.gov/sds/mochi-boot-camp/

SSG initialization

SSG initialization

```c
#include <margo.h>
#include <ssg-mpi.h>

int main(int argc, char** argv)
{
    MPI_Init(&argc, &argv);

    margo_instance_id mid = margo_init("tcp", MARGO_SERVER_MODE, 0, -1);
    assert(mid);

    ssg_init(mid);
    ...
    ssg_finalize();
    ...
    margo_wait_for_finalize(mid);
    MPI_Finalize();
    return 0;
}
```

https://xgitlab.cels.anl.gov/sds/mochi-boot-camp/

SSG initialization

```c
#include <margo.h>
#include <ssg-mpi.h>

int main(int argc, char** argv)
{
    MPI_Init(&argc, &argv);

    margo_instance_id mid = margo_init("tcp", MARGO_SERVER_MODE, 0, -1);
    assert(mid);

    ssg_init(mid);
    ...
    ssg_finalize();
    ...
    margo_wait_for_finalize(mid);
    MPI_Finalize();
    return 0;
}
```

Use MPI for bootstrapping

MARGO_SERVER_MODE required for all group members

Corresponding call to ssg_finalize() before shutting down server -- *ALL* SSG calls must be made between ssg_init and ssg_finalize()

https://xgitlab.cels.anl.gov/sds/mochi-boot-camp/

Creating groups

Creating groups using MPI communicator

```c
#include <margo.h>
#include <ssg-mpi.h>

int main(int argc, char** argv)
{
    ...
    ssg_group_id_t g_id;
    g_id = ssg_group_create_mpi("group-foo", MPI_COMM_WORLD, NULL, NULL);
    assert(g_id != SSG_GROUP_ID_INVALID);
    ...
    ssg_group_destroy(g_id);
    ...
    margo_wait_for_finalize(mid);
    MPI_Finalize();
    return 0;
}
```

https://xgitlab.cels.anl.gov/sds/mochi-boot-camp/

Creating groups using MPI communicator

```c
#include <margo.h>
#include <ssg-mpi.h>

int main(int argc, char** argv)
{
    ...
    ssg_group_id_t g_id;
    g_id = ssg_group_create_mpi("group-foo", MPI_COMM_WORLD, NULL, NULL);
    assert(g_id != SSG_GROUP_ID_INVALID);
    ...
    ssg_group_destroy(g_id);
    ...
    margo_wait_for_finalize(mid);
    MPI_Finalize();
    return 0;
```

Arguments allowing definition of a callback for any membership changes

g_id uniquely identifies group, used in subsequent calls for managing this group

Corresponding call to ssg_group_destroy() at some point before shutting down server

xgitlab.cels.anl.gov/sds/mochi-boot-camp/

Querying group state

Querying group state

```c
#include <margo.h>
#include <ssg-mpi.h>

int main(int argc, char** argv)
{
    ...
    ssg_group_id_t g_id;
    g_id = ssg_group_create_mpi("group-foo", MPI_COMM_WORLD, NULL, NULL);
    assert(g_id != SSG_GROUP_ID_INVALID);

    int self_rank;
    int group_size;
    self_rank = ssg_get_group_self_rank(g_id);
    assert(self_rank >= 0);
    group_size = ssg_get_group_size(g_id);
    assert(group_size > 0);
    ...
}
```

https://xgitlab.cels.anl.gov/sds/mochi-boot-camp/

Querying group state

```c
#include <margo.h>
#include <ssg-mpi.h>

int main(int argc, char** argv)
{
    ...
    ssg_group_id_t g_id;
    g_id = ssg_group_create_mpi("group-foo", MPI_COMM_WORLD,
    assert(g_id != SSG_GROUP_ID_INVALID);

    int self_rank;
    int group_size;
    self_rank = ssg_get_group_self_rank(g_id);
    assert(self_rank >= 0);
    group_size = ssg_get_group_size(g_id);
    assert(group_size > 0);
    ...
}
```

Obtain caller's rank in the created group. Note that SSG member IDs are unique across groups unlike ranks and can be obtained with ssg_get_self_id()

Total number of members in the group, including self if caller is a member (not observer)

https://xgitlab.cels.anl.gov/sds/mochi-boot-camp/

Querying group state

```c
#include <margo.h>
#include <ssg-mpi.h>

int main(int argc, char** argv)
{
    ...
    int member_rank;
    ssg_member_id_t member_id;
    hg_addr_t member_addr;

    member_id = ssg_get_group_member_id_from_rank(g_id, member_rank);
    assert(member_id != SSG_MEMBER_ID_INVALID);
    member_addr = ssg_get_group_member_addr(g_id, member_id);
    assert(member_addr != HG_ADDR_NULL);
    ...
}
```

https://xgitlab.cels.anl.gov/sds/mochi-boot-camp/

Querying group state

```c
#include <margo.h>
#include <ssg-mpi.h>

int main(int argc, char** argv)
{
    ...
    int member_rank;
    ssg_member_id_t member_id;
    hg_addr_t member_addr;

    member_id = ssg_get_group_member_id_from_rank(g_id, member_rank);
    assert(member_id != SSG_MEMBER_ID_INVALID);
    member_addr = ssg_get_group_member_addr(g_id, member_id);
    assert(member_addr != HG_ADDR_NULL);
    ...
}
```

Translate group member rank into SSG member ID so we can query its state

Using the member's ID, retrieve its Mercury address so we can subsequently send RPCs to it

https://xgitlab.cels.anl.gov/sds/mochi-boot-camp/

Sharing group info

Sharing group info

member

```c
int main(int argc, char** argv)
{
    ...
    ssg_group_id_t g_id;
    g_id = ssg_group_create_mpi("group-foo", MPI_COMM_WORLD, NULL, NULL);
    assert(g_id != SSG_GROUP_ID_INVALID);

    ssg_group_id_store("/tmp/gid_file", g_id);
    ...
}
```

non-member

```c
int main(int argc, char** argv)
{
    ...
    ssg_group_id_t g_id;
    ssg_group_id_load("/tmp/gid_file", &g_id);
    ...
}
```

https://xgitlab.cels.anl.gov/sds/mochi-boot-camp/

Sharing group info

member

```c
int main(int argc, char** argv)
{
    ...
    ssg_group_id_t g_id;
    g_id = ssg_group_create_mpi("group-foo", MPI_COMM_WORLD, NULL, NULL);
    assert(g_id != SSG_GROUP_ID_INVALID);

    ssg_group_id_store("/tmp/gid_file", g_id);
    ...
}
```

SSG also has generic group ID serialization functions, so users can share using MPI, PMIx, kv, etc

non-member

```c
int main(int argc, char** argv)
{
    ...
    ssg_group_id_t g_id;
    ssg_group_id_load("/tmp/gid_file", &g_id);
    ...
}
```

After loading, SSG maintains minimal state on group until it is joined, observed, or destroyed

https://xgitlab.cels.anl.gov/sds/mochi-boot-camp/

Observing groups

Observing groups

```c
#include <margo.h>
#include <ssg-mpi.h>

int main(int argc, char** argv)
{
    margo_instance_id mid = margo_init("tcp", MARGO_CLIENT_MODE, 0, -1);
    assert(mid);
    ...
    ssg_group_id_t g_id;
    ssg_group_id_load("/tmp/gid_file", &g_id);

    ssg_group_observe(g_id);
    ...
    ssg_group_unobserve(g_id);
    ...
}
```

https://xgitlab.cels.anl.gov/sds/mochi-boot-camp/

Observing groups

```c
#include <margo.h>
#include <ssg-mpi.h>

int main(int argc, char** argv)
{
    margo_instance_id mid = margo_init("tcp", MARGO_CLIENT_MODE, 0, -1);
    assert(mid);
    ...
    ssg_group_id_t g_id;
    ssg_group_id_load("/tmp/gid_file", &g_id);

    ssg_group_observe(g_id);
    ...
    ssg_group_unobserve(g_id);
    ...
}
```

MARGO_SERVER_MODE is not required for observing a group

Observing a group allows a client to access membership state without actively participating in the group

https://xgitlab.cels.anl.gov/sds/mochi-boot-camp/

Dynamically joining/leaving groups

Dynamically joining/leaving groups

```c
#include <margo.h>
#include <ssg-mpi.h>

int main(int argc, char** argv)
{
    margo_instance_id mid = margo_init("tcp", MARGO_SERVER_MODE, 0, -1);
    ...
    ssg_group_id_t g_id;
    ssg_group_id_load("/tmp/gid_file", &g_id);

    ssg_group_join(g_id);
    ...
    ssg_group_leave(g_id);
    ...
}
```

https://xgitlab.cels.anl.gov/sds/mochi-boot-camp/

Dynamically joining/leaving groups

```c
#include <margo.h>
#include <ssg-mpi.h>

int main(int argc, char** argv)
{
    margo_instance_id mid = margo_init("tcp", MARGO_SERVER_MODE, 0, -1);
    ...
    ssg_group_id_t g_id;
    ssg_group_id_load("/tmp/gid_file", &g_id);

    ssg_group_join(g_id);
    ...
    ssg_group_leave(g_id);
    ...
}
```

MARGO_SERVER_MODE required to join

After joining, other group members will maintain connection with this process

Any member can leave at any time, and other processes will eventually learn of this

https://xgitlab.cels.anl.gov/sds/mochi-boot-camp/

Detecting group member failures

Detecting group member failures

```
#include <margo.h>
#include <ssg-mpi.h>

void ssg_membership_update_cb(void *g_data, ssg_member_id_t member,
    ssg_membership_update_t update_type)
{
    if((update_type == SSG_MEMBER_DIED) || (update_type == SSG_MEMBER_LEFT))
        printf("member %lu left group %lu\n", member, *(ssg_group_id_t *)g_data);
    else
        printf("member %lu joined group %lu\n", member, *(ssg_group_id_t *)g_data);
}
int main(int argc, char** argv)
{
    ...
    ssg_group_id_t g_id;
    g_id = ssg_group_create_mpi("group-foo", MPI_COMM_WORLD, ssg_membership_update_cb, &g_id);
    assert(g_id != SSG_GROUP_ID_INVALID);
    ...
}
```

https://xgitlab.cels.anl.gov/sds/mochi-boot-camp/

Detecting group member failures

```
#include <margo.h>
#include <ssg-mpi.h>

void ssg_membership_update_cb(void *g_data, ssg_membe
    ssg_membership_update_t update_type)
{
    if((update_type == SSG_MEMBER_DIED) || (update_type == SSG_MEMBER_LEFT))
        printf("member %lu left group %lu\n", member, *(ssg_group_id_t *)g_data);
    else
        printf("member %lu joined group %lu\n", member, *(ssg_group_id_t *)g_data);
}
int main(int argc, char** argv)
{
    ...
    ssg_group_id_t g_id;
    g_id = ssg_group_create_mpi("group-foo", MPI_COMM_WORLD, ssg_membership_update_cb, &g_id);
    assert(g_id != SSG_GROUP_ID_INVALID);
    ...
}
```

3 potential updates for a group member: DIED (eviction by failure detector), LEFT (explicit leave), JOINED

Provide callback for notification on group membership changes

https://xgitlab.cels.anl.gov/sds/mochi

SSG failure detection

Failure detection is on all the time for all SSG groups (members only), using multiple detection mechanisms:

- SWIM, a gossip-based group membership protocol, is enabled on all groups
  - Processes periodically probe other processes for liveness
  - Processes gossip about perceived state of other processes to reach eventual consensus
  - Numerous tunables to control detection latency, accuracy, and network load
  - We have modified SWIM to help implement dynamic leaves/joins in SSG
- (on applicable systems) PMIx event notification system
  - Register for event notifications from the RM regarding potential process or system failures

SSG failures (and explicit leaves) are currently irreversible!

[1] A. Das, I. Gupta, & A. Motivala. “SWIM: Scalable Weakly-consistent Infection-style Process Group Membership Protocol”

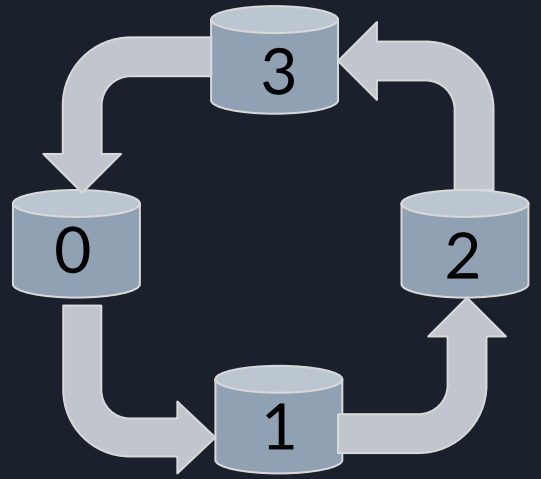
https://xgitlab.cels.anl.gov/sds/mochi-boot-camp/

SSG exercise: token ring network

SSG exercise: token ring network

Using SSG rank information, create a logical ring network topology and forward a token along it, starting at rank 0 (i.e., 0->1->...->N->0)

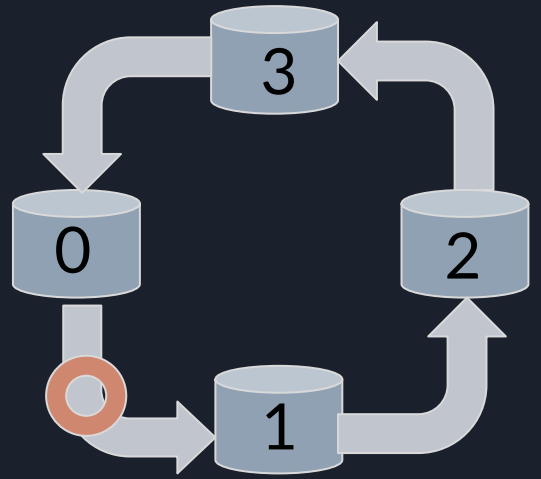
After each rank receives the token, it shuts down



https://xgitlab.cels.anl.gov/sds/mochi-boot-camp/

SSG exercise: token ring network

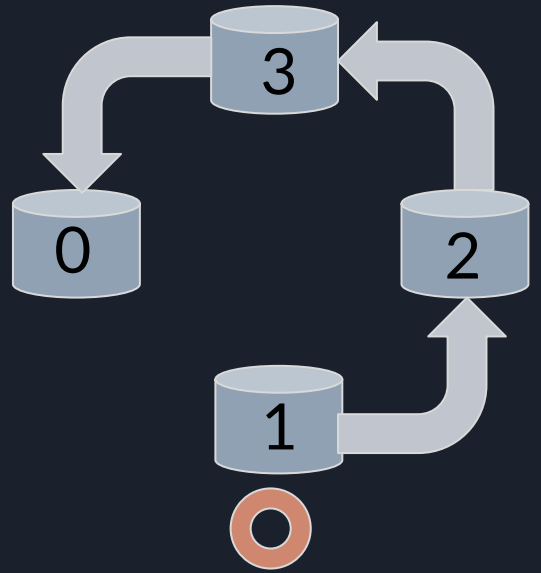
```
Member 0 forwarding token 48879 to 1
Member 1 got token 48879
Member 1 forwarding token 48879 to 2
Member 1 shutting down
Member 2 got token 48879
Member 2 forwarding token 48879 to 3
Member 2 shutting down
Member 3 got token 48879
Member 3 forwarding token 48879 to 0
Member 3 shutting down
Member 0 got token 48879
Member 0 shutting down
```



https://xgitlab.cels.anl.gov/sds/mochi-boot-camp/

SSG exercise: token ring network

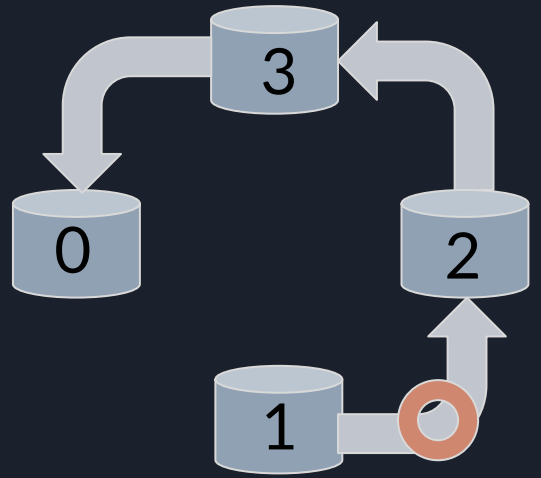
```
Member 0 forwarding token 48879 to 1
Member 1 got token 48879
Member 1 forwarding token 48879 to 2
Member 1 shutting down
Member 2 got token 48879
Member 2 forwarding token 48879 to 3
Member 2 shutting down
Member 3 got token 48879
Member 3 forwarding token 48879 to 0
Member 3 shutting down
Member 0 got token 48879
Member 0 shutting down
```



https://xgitlab.cels.anl.gov/sds/mochi-boot-camp/

SSG exercise: token ring network

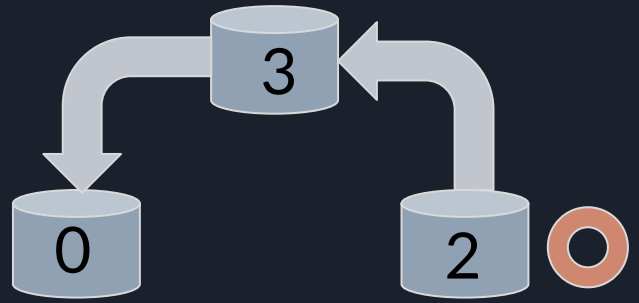
```
Member 0 forwarding token 48879 to 1
Member 1 got token 48879
Member 1 forwarding token 48879 to 2
Member 1 shutting down
Member 2 got token 48879
Member 2 forwarding token 48879 to 3
Member 2 shutting down
Member 3 got token 48879
Member 3 forwarding token 48879 to 0
Member 3 shutting down
Member 0 got token 48879
Member 0 shutting down
```



https://xgitlab.cels.anl.gov/sds/mochi-boot-camp/

SSG exercise: token ring network

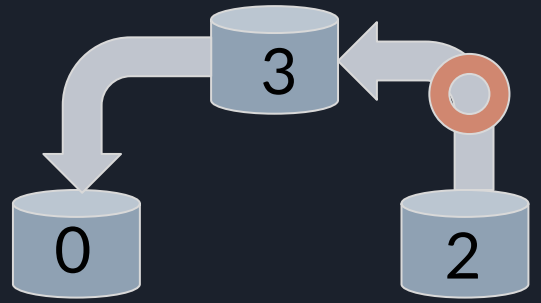
```
Member 0 forwarding token 48879 to 1
Member 1 got token 48879
Member 1 forwarding token 48879 to 2
Member 1 shutting down
Member 2 got token 48879
Member 2 forwarding token 48879 to 3
Member 2 shutting down
Member 3 got token 48879
Member 3 forwarding token 48879 to 0
Member 3 shutting down
Member 0 got token 48879
Member 0 shutting down
```



https://xgitlab.cels.anl.gov/sds/mochi-boot-camp/

SSG exercise: token ring network

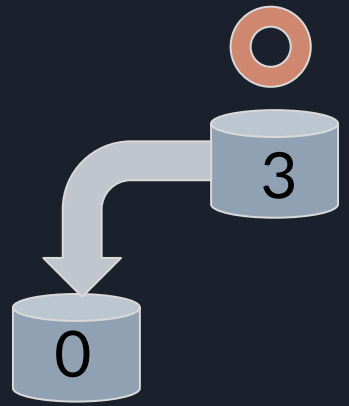
```
Member 0 forwarding token 48879 to 1
Member 1 got token 48879
Member 1 forwarding token 48879 to 2
Member 1 shutting down
Member 2 forwarding token 48879 to 3
Member 2 shutting down
Member 3 forwarding token 48879 to 0
Member 3 shutting down
Member 0 got token 48879
Member 0 shutting down
```



https://xgitlab.cels.anl.gov/sds/mochi-boot-camp/

SSG exercise: token ring network

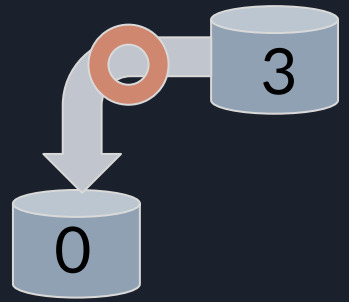
```
Member 0 forwarding token 48879 to 1
Member 1 got token 48879
Member 1 forwarding token 48879 to 2
Member 1 shutting down
Member 2 got token 48879
Member 2 forwarding token 48879 to 3
Member 2 shutting down
Member 3 got token 48879
Member 3 forwarding token 48879 to 0
Member 3 shutting down
Member 0 got token 48879
Member 0 shutting down
```



https://xgitlab.cels.anl.gov/sds/mochi-boot-camp/

SSG exercise: token ring network

```
Member 0 forwarding token 48879 to 1
Member 1 got token 48879
Member 1 forwarding token 48879 to 2
Member 1 shutting down
Member 2 got token 48879
Member 2 forwarding token 48879 to 3
Member 2 shutting down
Member 3 got token 48879
Member 3 forwarding token 48879 to 0
Member 3 shutting down
Member 0 got token 48879
Member 0 shutting down
```



https://xgitlab.cels.anl.gov/sds/mochi-boot-camp/

SSG exercise: token ring network

```
Member 0 forwarding token 48879 to 1
Member 1 got token 48879
Member 1 forwarding token 48879 to 2
Member 1 shutting down
Member 2 got token 48879
Member 2 forwarding token 48879 to 3
Member 2 shutting down
Member 3 got token 48879
Member 3 shutting down
Member 0 got token 48879
```

https://xgitlab.cels.anl.gov/sds/mochi-boot-camp/

SSG exercise: token ring network

```
Member 0 forwarding token 48879 to 1
Member 1 got token 48879
Member 1 forwarding token 48879 to 2
Member 1 shutting down
Member 2 got token 48879
Member 2 forwarding token 48879 to 3
Member 2 shutting down
Member 3 got token 48879
Member 3 forwarding token 48879 to 0
Member 3 shutting down
Member 0 got token 48879
Member 0 shutting down
```

https://xgitlab.cels.anl.gov/sds/mochi-boot-camp/

Example solution

Server state

```
struct server_data
{
    margo_instance_id mid;
    ssg_group_id_t gid;
    int self_rank;
    int group_size;
    hg_id_t token_forward_rpc_id;
};
```

https://xgitlab.cels.anl.gov/sds/mochi-boot-camp/

Example solution

Server state

Margo and SSG group state needed inside of RPC handlers

```
struct server_data
{
    margo_instance_id mid;
    ssg_group_id_t gid;
    int self_rank;
    int group_size;
    hg_id_t token_forward_rpc_id;
};
```

https://xgitlab.cels.anl.gov/sds/mochi-boot-camp/

Example solution

Initialization

```
int main(int argc, char** argv)
{
    struct server_data serv_data;

    MPI_Init(&argc, &argv);

    serv_data.mid = margo_init("na+sm", MARGO_SERVER_MODE, 0, -1);
    assert(serv_data.mid);

    ssg_init(serv_data.mid);
    ...

}
```

https://xgitlab.cels.anl.gov/sds/mochi-boot-camp/

Example solution

Initialization

```
int main(int argc, char** argv)
{
    struct server_data serv_data;

    MPI_Init(&argc, &argv);

    serv_data.mid = margo_init("na+sm", MARGO_SERVER_MODE, 0, -1);
    assert(serv_data.mid);

    ssg_init(serv_data.mid);
    ...

}
```

Server state structure for sharing state with RPC handlers

MPI for bootstrapping

Shared memory plugin for communication

https://xgitlab.cels.anl.gov/sds/mochi-boot-camp/

Example solution

RPC registration

```
MERCURY_GEN_PROC(token_t,
    ((uint32_t)(token)))

static void token_forward_recv(hg_handle_t handle);
DECLARE_MARGO_RPC_HANDLER(token_forward_recv)

{
   …
   serv_data.token_forward_rpc_id = MARGO_REGISTER(serv_data.mid, "token_forward",
        token_t, void, token_forward_recv);
    margo_registered_disable_response(serv_data.mid, serv_data.token_forward_rpc_id,
        HG_TRUE);
    margo_register_data(serv_data.mid, serv_data.token_forward_rpc_id, &serv_data, NULL);
    ...
}
```

https://xgitlab.cels.anl.gov/sds/mochi-boot-camp/

Example solution

RPC registration

Simple token type and serialization macro

Forward declare token receive RPC handlers

Register RPC, providing input/output types and handler name

Enable 1-way RPCs and register our server_data structure with the handler

```
MERCURY_GEN_PROC(token_t,
    ((uint32_t)(token)))

static void token_forward_recv(hg_handle_t handle);
DECLARE_MARGO_RPC_HANDLER(token_forward_recv)

{
   …
   serv_data.token_forward_rpc_id = MARGO_REGISTER(serv_data.mid, "token_forward",
       token_t, void, token_forward_recv);
    margo_registered_disable_response(serv_data.mid, serv_data.token_forward_rpc_id,
       HG_TRUE);
    margo_register_data(serv_data.mid, serv_data.token_forward_rpc_id, &serv_data, NULL);
    ...
}
```

https://xgitlab.cels.anl.gov/sds/mochi-b

Example solution

Group creation

```
{
    ...
    serv_data.gid = ssg_group_create_mpi("token-ring-group", MPI_COMM_WORLD, NULL, NULL);
    assert(serv_data.gid != SSG_GROUP_ID_INVALID);

    serv_data.self_rank = ssg_get_group_self_rank(serv_data.gid);
    assert(serv_data.self_rank >= 0);
    serv_data.group_size = ssg_get_group_size(serv_data.gid);
    assert(serv_data.group_size > 0);
    ...
}
```

https://xgitlab.cels.anl.gov/sds/mochi-boot-camp/

Example solution

Group creation

MPI group creation function using MPI_COMM_WORLD

```
{
    ...
    serv_data.gid = ssg_group_create_mpi("token-ring-group", MPI_COMM_WORLD, NULL, NULL);
    assert(serv_data.gid != SSG_GROUP_ID_INVALID);

    serv_data.self_rank = ssg_get_group_self_rank(serv_data.gid);
    assert(serv_data.self_rank >= 0);
    serv_data.group_size = ssg_get_group_size(serv_data.gid);
    assert(serv_data.group_size > 0);
    ...
}
```

Retrieve group rank and size using SSG, this is needed to implement token ring

https://xgitlab.cels.anl.gov/sds/mochi-boot-camp/

Example solution

Token forwarding kickoff

```
void token_forward(struct server_data *serv_data);

{
    ...
    if (serv_data.self_rank == 0)
        token_forward(&serv_data);

    margo_wait_for_finalize(serv_data.mid);
    MPI_Finalize();
    return 0;
}
```

https://xgitlab.cels.anl.gov/sds/mochi-boot-camp/

Example solution

Token forwarding kickoff

Forward declare function for forwarding the token to the next rank

```
void token_forward(struct server_data *serv_data);

{
    ...
    if (serv_data.self_rank == 0)
        token_forward(&serv_data);

    margo_wait_for_finalize(serv_data.mid);
    MPI_Finalize();
    return 0;
}
```

Rank 0 initiates the token forwarding,

All ranks wait for a finalize signal before exiting

https://xgitlab.cels.anl.gov/sds/mochi-boot-camp/

Example solution

Token forwarding

```
void token_forward(struct server_data *serv_data)
{
    int target_rank = (serv_data->self_rank + 1) % serv_data->group_size;

    ssg_member_id_t target_id = ssg_get_group_member_id_from_rank(
        serv_data->gid, target_rank);

    hg_addr_t target_addr = ssg_get_group_member_addr(serv_data->gid, target_id);
    ...
}
```

https://xgitlab.cels.anl.gov/sds/mochi-boot-camp/

Example solution

Token forwarding

```
void token_forward(struct server_data *serv_data)
{
    int target_rank = (serv_data->self_rank + 1) % serv_data->group_size;

    ssg_member_id_t target_id = ssg_get_group_member_id_from_rank(
        serv_data->gid, target_rank);

    hg_addr_t target_addr = ssg_get_group_member_addr(serv_data->g    rget_id);
    ...
}
```

Use self_rank, group_size, and module to determine target

Convert rank to SSG member ID

Use SSG to determine Mercury address of target

https://xgitlab.cels.anl.gov/sds/mochi-boot-camp/

Example solution

Token forwarding

```
void token_forward(struct server_data *serv_data)
{
    ...
    hg_handle_t h;
    token_t fwd_token;

    printf("Member %d forwarding token %u to %d\n",
        serv_data->self_rank, fwd_token.token, target_rank);
    fwd_token.token = 0xBEEF;
    margo_create(serv_data->mid, target_addr, serv_data->token_forward_rpc_id, &h);
    margo_forward(h, &fwd_token);
    margo_destroy(h);
}
```

https://xgitlab.cels.anl.gov/sds/mochi-boot-camp/

Example solution

Token forwarding

```
void token_forward(struct server_data *serv_data)
{
    ...
    hg_handle_t h;
    token_t fwd_token;

    printf("Member %d forwarding token %u to %d\n",
        serv_data->self_rank, fwd_token.token, target_rank);
    fwd_token.token = 0xBEEF;
    margo_create(serv_data->mid, target_addr, serv_data->token_forward_rpc_id, &h);
    margo_forward(h, &fwd_token);
    margo_destroy(h);
}
```

Set token value

Create token handle and forward to target, then destroy the handle

https://xgitlab.cels.anl.gov/sds/mochi-boot-camp/

Example solution

Token receive handler

```
static void token_forward_recv(hg_handle_t h)
{
    token_t fwd_token;

    margo_instance_id mid = margo_hg_handle_get_instance(h);
    const struct hg_info* info = margo_get_info(h);
    struct server_data* serv_data = (struct server_data *)
        margo_registered_data(mid, info->id);

    margo_get_input(h, &fwd_token);
    printf("Member %d got token %u\n", serv_data->self_rank, fwd_token.token);
    margo_free_input(h, &fwd_token);
    ...
}
DEFINE_MARGO_RPC_HANDLER(token_forward_recv)
```

https://xgitlab.cels.anl.gov/sds/mochi-boot-camp/

Example solution

Token receive handler

```
static void token_forward_recv(hg_handle_t h)
{
    token_t fwd_token;

    margo_instance_id mid = margo_hg_handle_get_instance(h);
    const struct hg_info* info = margo_get_info(h);
    struct server_data* serv_data = (struct server
        margo_registered_data(mid, info->id);

    margo_get_input(h, &fwd_token);
    printf("Member %d got token %u\n", serv_data
    margo_free_input(h, &fwd_token);
    ...
}
DEFINE_MARGO_RPC_HANDLER(token_forward_recv)
```

Use Mercury handle to retrieve the server data structure we registered with this handler

Get the input token and print to confirm value -- don't forget to free your inputs or outputs!

Use MARGO RPC handler definition macro to setup proper wrappers

https://xgitlab.cels.anl.gov/sds/mochi-boot-camp/

Example solution

Token receive handler

```
static void token_forward_recv(hg_handle_t h)
{
    ...
    if (serv_data->self_rank > 0)
        token_forward(serv_data);

    printf("Member %d shutting down\n", serv_data->self_rank);

    ssg_group_destroy(serv_data->gid);
    ssg_finalize();
    margo_finalize(serv_data->mid);
}
DEFINE_MARGO_RPC_HANDLER(token_forward_recv)
```

https://xgitlab.cels.anl.gov/sds/mochi-boot-camp/

Example solution

Token receive handler

```
static void token_forward_recv(hg_handl
{
    ...
    if (serv_data->self_rank > 0)
        token_forward(serv_data);

    printf("Member %d shutting down\n", s

    ssg_group_destroy(serv_data->gid);
    ssg_finalize();
    margo_finalize(serv_data->mid);
}
DEFINE_MARGO_RPC_HANDLER(token_forward_recv)
```

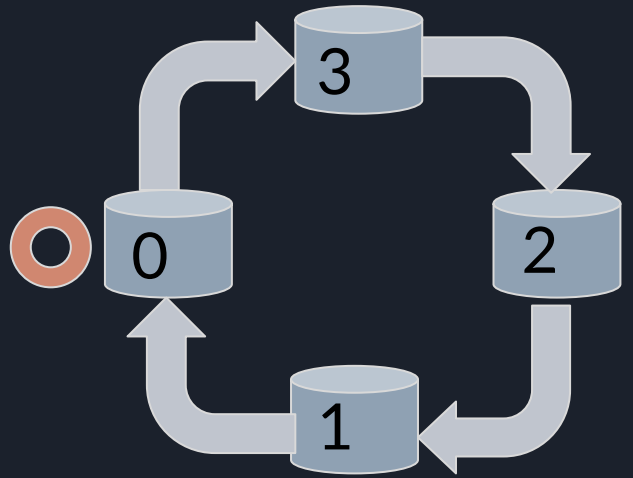
Non-zero ranks continue to forward the token, rank 0 stops

Signal finalize so this rank can shut down

https://xgitlab.cels.anl.gov/sds/mochi-boot-camp/

SSG exercise: token ring network

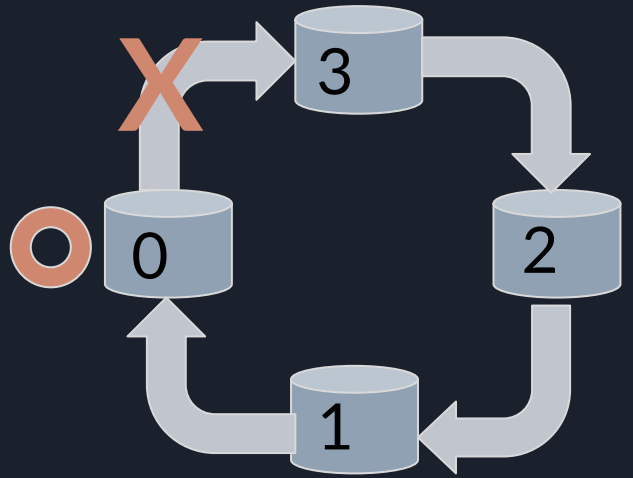
Now, extend the example to have servers remain running after receiving the token, with rank 0 sending a shutdown signal through the ring in reverse order (i.e., rank 3 shuts down first, rank 0 shuts down last



https://xgitlab.cels.anl.gov/sds/mochi-boot-camp/

SSG exercise: token ring network

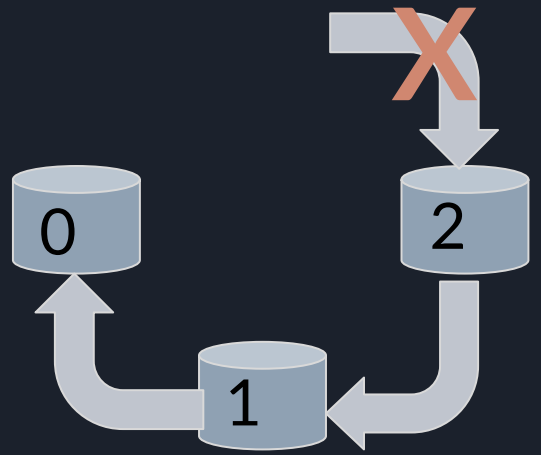
```
Member 0 got token 48879
Member 0 forwarding shutdown to 3
Member 3 shutting down
Member 2 forwarding shutdown to 1
Member 2 shutting down
Member 1 forwarding shutdown to 0
Member 1 shutting down
Member 0 shutting down
```



https://xgitlab.cels.anl.gov/sds/mochi-boot-camp/

SSG exercise: token ring network

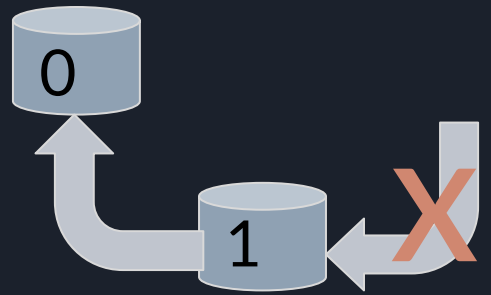
```
...
Member 0 got token 48879
Member 3 forwarding shutdown to 2
Member 3 shutting down
Member 2 shutting down
Member 1 forwarding shutdown to 0
Member 1 shutting down
Member 0 shutting down
```



https://xgitlab.cels.anl.gov/sds/mochi-boot-camp/

SSG exercise: token ring network

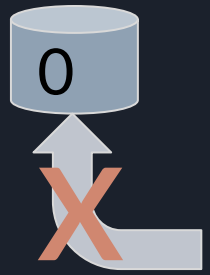
```
...
Member 0 got token 48879
Member 0 forwarding shutdown to 3
Member 3 forwarding shutdown to 2
Member 2 forwarding shutdown to 1
Member 2 shutting down
Member 1 shutting down
Member 0 shutting down
```



https://xgitlab.cels.anl.gov/sds/mochi-boot-camp/

SSG exercise: token ring network

```
...
Member 0 got token 48879
Member 0 forwarding shutdown to 3
Member 3 forwarding shutdown to 2
Member 3 shutting down
Member 2 forwarding shutdown to 1
Member 1 forwarding shutdown to 0
Member 1 shutting down
```



https://xgitlab.cels.anl.gov/sds/mochi-boot-camp/

SSG exercise: token ring network

```
...
Member 0 got token 48879
Member 0 forwarding shutdown to 3
Member 3 forwarding shutdown to 2
Member 3 shutting down
Member 2 forwarding shutdown to 1
Member 2 shutting down
Member 1 forwarding shutdown to 0
Member 1 shutting down
Member 0 shutting down
```

https://xgitlab.cels.anl.gov/sds/mochi-boot-camp/

Example solution

Server state

```
struct server_data
{
    margo_instance_id mid;
    ssg_group_id_t gid;
    int self_rank;
    int group_size;
    hg_id_t token_forward_rpc_id;
    hg_id_t shutdown_forward_rpc_id;
};
```

https://xgitlab.cels.anl.gov/sds/mochi-boot-camp/

Example solution

Server state

```
struct server_data
{
    margo_instance_id mid;
    ssg_group_id_t gid;
    int self_rank;
    int group_size;
    hg_id_t token_forward_rpc_id;
    hg_id_t shutdown_forward_rpc_id;
};
```

Add shutdown forward RPC ID

https://xgitlab.cels.anl.gov/sds/mochi-boot-camp/

Example solution

RPC registration

```
static void shutdown_forward_recv(hg_handle_t handle);
DECLARE_MARGO_RPC_HANDLER(shutdown_forward_recv)

{
   …
   serv_data.shutdown_forward_rpc_id = MARGO_REGISTER(serv_data.mid, "shutdown_forward",
       void, void, shutdown_forward_recv);
    margo_registered_disable_response(serv_data.mid, serv_data.shutdown_forward_rpc_id,
       HG_TRUE);
    margo_register_data(serv_data.mid, serv_data.shutdown_forward_rpc_id, &serv_data, NULL);
    ...
}
```

https://xgitlab.cels.anl.gov/sds/mochi-boot-camp/

Example solution

RPC registration

```
static void shutdown_forward_recv(hg_handle_t handle);
DECLARE_MARGO_RPC_HANDLER(shutdown_forward_recv)

{
  …
  serv_data.shutdown_forward_rpc_id = MARGO_REGISTER(serv_data.mid, "shutdown_forward",
      void, void, shutdown_forward_recv);
   margo_registered_disable_response(serv_data.mid, serv_data.shutdown_forward_rpc_id,
      HG_TRUE);
   margo_register_data(serv_data.mid, serv_data.shutdown_forward_rpc_id, &serv_data, NULL);
   ...
}
```

Forward declare shutdown receive RPC handlers

Register RPC, note that there is no input or output type for shutdown

Enable 1-way RPCs and register our server_data structure with the handler

https://xgitlab.cels.anl.gov/sds/mochi-boot-camp/

Example solution

Token receive handler

```
void shutdown_forward(struct server_data *serv_data);

static void token_forward_recv(hg_handle_t h)
{
    ...
    if (serv_data->self_rank > 0)
        token_forward(serv_data);
    else
        shutdown_forward(serv_data);
}
DEFINE_MARGO_RPC_HANDLER(token_forward_recv)
```

https://xgitlab.cels.anl.gov/sds/mochi-boot-camp/

Example solution

Token receive handler

```
void shutdown_forward(struct server_data *serv_data);

static void token_forward_recv(hg_handle_t h)
{
    ...
    if (serv_data->self_rank > 0)
        token_forward(serv_data);
    else
        shutdown_forward(serv_data);
}
DEFINE_MARGO_RPC_HANDLER(token_forward_recv)
```

Modify token receive logic so that rank 0 forwards a shutdown request on receipt

https://xgitlab.cels.anl.gov/sds/mochi-boot-camp/

Example solution

Shutdown forwarding

```
void shutdown_forward(struct server_data *serv_data)
{
    int target_rank = (serv_data->self_rank - 1 + serv_data->group_size) %
        serv_data->group_size;

    ssg_member_id_t target_id = ssg_get_group_member_id_from_rank(
        serv_data->gid, target_rank);

    hg_addr_t target_addr = ssg_get_group_member_addr(serv_data->gid, target_id);
    ...
}
```

https://xgitlab.cels.anl.gov/sds/mochi-boot-camp/

Example solution

Shutdown forwarding

```
void shutdown_forward(struct server_data *serv_data)
{
    int target_rank = (serv_data->self_rank - 1 + serv_data->group_size) %
        serv_data->group_size;

    ssg_member_id_t target_id = ssg_get_group_member_id_from_rank(
        serv_data->gid, target_rank);

    hg_addr_t target_addr = ssg_get_group_member_addr(serv_data
    ...
}
```

Use self_rank, group_size, and module to determine target. Note we are going in reverse rank order

Convert rank to SSG member ID

Use SSG to determine Mercury address of target

https://xgitlab.cels.anl.gov/sds/mochi-boot-camp/

Example solution

Shutdown forwarding

```
void shutdown_forward(struct server_data *serv_data)
{
    ...
    hg_handle_t h;

    printf("Member %d forwarding shutdown to %d\n",
        serv_data->self_rank, target_rank);
    margo_create(serv_data->mid, target_addr, serv_data->shutdown_forward_rpc_id, &h);
    margo_forward(h, NULL);
    margo_destroy(h);
}
```

https://xgitlab.cels.anl.gov/sds/mochi-boot-camp/

Example solution

Shutdown forwarding

```
void shutdown_forward(struct server_data *serv_data)
{
    ...
    hg_handle_t h;

    printf("Member %d forwarding shutdown to %d\n",
        serv_data->self_rank, target_rank);
    margo_create(serv_data->mid, target_addr, serv_data->shutdown_forward_rpc_id, &h);
    margo_forward(h, NULL);
    margo_destroy(h);
}
```

Create shutdown handle and forward to target, then destroy the handle. Note NULL input to forward RPC

https://xgitlab.cels.anl.gov/sds/mochi-boot-camp/

Example solution

Shutdown receive handler

```
static void shutdown_forward_recv(hg_handle_t h)
{
    margo_instance_id mid = margo_hg_handle_get_instance(h);
    const struct hg_info* info = margo_get_info(h);
    struct server_data* serv_data = (struct server_data *)
        margo_registered_data(mid, info->id);

    if (serv_data->self_rank > 0)
        shutdown_forward(serv_data);

    printf("Member %d shutting down\n", serv_data->self_rank);
    margo_destroy(h);
    ssg_group_destroy(serv_data->gid);
    ssg_finalize();
    margo_finalize(serv_data->mid);
}
DEFINE_MARGO_RPC_HANDLER(shutdown_forward_recv)
```

https://xgitlab.cels.anl.gov/sds/mochi-boot-camp/

Example solution

Shutdown receive handler

```
static void shutdown_forward_recv(hg_handle_t h)
{
    margo_instance_id mid = margo_hg_handle_get_instanc (h);
    const struct hg_info* info = margo_get_info(h);
    struct server_data* serv_data = (struct server_data
        margo_registered_data(mid, info->id);

    if (serv_data->self_rank > 0)
        shutdown_forward(serv_data);

    printf("Member %d shutting down\n", serv_data->sel
    margo_destroy(h);
    ssg_group_destroy(serv_data->gid);
    ssg_finalize();
    margo_finalize(serv_data->mid);
}
DEFINE_MARGO_RPC_HANDLER(shutdown_forward_recv)
```

Use Mercury handle to retrieve the server data structure we registered with this handler

Non-zero ranks continue to forward the shutdown, rank 0 stops

Signal finalize so this rank can shut down

Use MARGO RPC handler definition macro to setup proper wrappers

https://xgitlab.cels.anl.gov/sds/moc