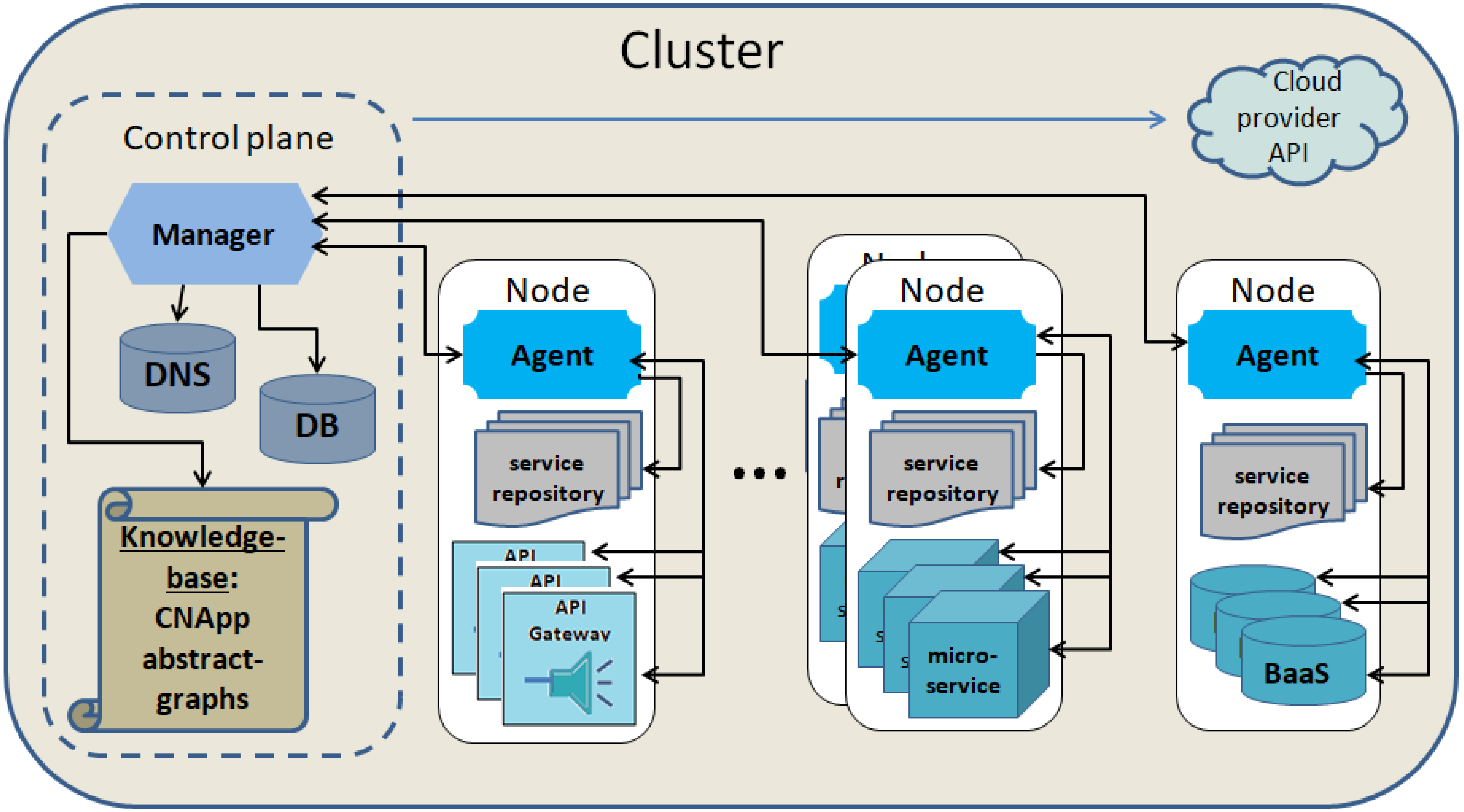
SSMMP


Cluster
Control plane
Cloud provider API
Manager
DNS
DB
Knowledge-base: CNApp abstract-graphs
Node
Agent
service repository
API
API Gateway
micro-service
BaaS
• • •

SSMMP - specification of the protocol messages and actions

- normal (not italic) letters to denote services, their instances, plugs, sockets and connections, like A, P, S, B, i, and j.
- There are two general kids of messages: request, and response to this request.
- All messages are strings.
- Message consists of a sequence of lines.
- Line is of the form:

```
line_name: contents
```

- The first line is for message type.
- The second line is for message identifier (an integer). The identifier is unique, and is the same for request and its response.

Initialization of the protocol

- Let a CNApp be fixed. The abstract graph of CNApp is known to Manager. For each service of CNApp, there are agents that can execute instances of this service, i.e. the bytecode of this service belongs to the agents repositories.

- Each entry of the repository is of the form (service-name, list of socket names, list of plug names, bytecode of service).

- An agent registers with Manager and sends the list of service names of its repository.
- Request for the registration from agent to Manager is of the following form.

Request for the registration from agent to Manager

```
type: initiation_request
message_id: [integer]
agent_network_address: [IPv6]
service_repository: [service name list]
```

In square brackets of

```
      message_id: [integer]
```

there is an element of a datatype, in this case it is a positive integer determined by the agent.

For `agent_network_address: [IPv6]`

it is a concrete IPv6 network address.

For the line

```
      service_repository: [service name list]
```

the contents denotes a sequence of the form

```
      (name1; name2; … name_k)
```

Registration response form Manager to agent

```
type: initiation_response
message_id: [integer]
status: [status code]
```

- Universal HTTP response status codes are proposed to be adapted to SSMMP. Their meaning depends on response types.
- Each code consists of three digits, and is of the form:

- `1xx informational response`
- `2xx successful` – the request was successfully received, understood, and accepted
- `3xx redirection` – further action needs to be taken in order to complete the request
- `4xx requester error` – the request contains bad syntax or cannot be fulfilled
- `5xx respondent error`

Execution of service A

- Manager assigns a unique identifier, say i, (a positive integer) to a new instance of A to be executed. Manager also determines port numbers to all sockets of the instance. It is called socket configuration, and is a sequence of pairs:

```
(socket_name, port_number)
```

- Configuration of plugs is a sequence of pairs:

```
(plug_name, service_name)
```

- assigned to instance i by Manager according to the CNApp abstract graph.
- This means that each abstract plug is assigned a service name, where is the corresponding abstract socket.
- Manager also determines a network address (denoted `NA_i` of the node where the instance i of A is to be executed by the agent residing on that node. % It is also the network address of the agent.

Request from Manager to the agent to execute the instance i of service A

```
type: execution_request
message_id: n
agent_network_address: NA_i
service_name: A
service_instance_id: i
socket_configuration: [configuration of sockets]
plug_configuration: [configuration of plugs]
```

- Action of the agent: execution of instance i of the service A for these configurations of sockets and plugs.
- Response from agent to Manager

```
type: execution_response
message_id: n
status: [status code]
```

Communication session establishment

- Establishing a communication session for abstract connection (A, (P,S), B) between instance i of A, and instance j of B.

- Let us suppose that instance i of service A is already running on the node that has network address `NA_i`

- Request from the instance i of service A to its agent:

```
type: session_request
message_id: n
sub_type: service_to_agent
source_service_name: A
source_service_instance_id: i
source_plug_name: P
dest_service_name: B
dest_socket_name: S
```

Communication session establishment

Request is forwarded to Manager by the agent:

```
type: session_request
message_id: n
sub_type: agent_to_Manager
agent_network_address: NA_i
source_service_name: A
source_service_instance_id: i
source_plug_name: P
dest_service_name: B
dest_socket_name: S
```

Communication session establishment

- If there is no instance of service B already running, then
- Manager sends a request to an agent to execute an instance j of service B.

- Otherwise, i.e. if instance j of service B (on the node with network address `NA_j` and the port k for S) is already running, then Manager sends the following response to the agent:

```
type: session_response
message_id: n
sub_type: Manager_to_agent
status: [status code]
dest_service_instance_network_address: NA_j
dest_socket_port: k
```

Communication session establishment

- Response from agent to instance of A:

```
type: session_response
message_id: n
sub_type: agent_to_service
status: [status code]
dest_service_instance_network_address: NA_j
dest_socket_port: k
```

Communication session establishment

- Action of instance i of A: initialize (P,S) session to instance j of B. The port number of plug P is determined; let it be denoted by m.
- By default, the socket S of instance j of B accepts the session establishment. This acceptance will be known to Manager, if the session acknowledgment is send by instance i to Manager via the agent.
- Action of instance j of B: accept the establishing (P,S) session to instance i of A.
- New socket port number (say l) is assigned to this session; this is exactly the same as for TCP connection.
- Instance j of B gets to know the values of the parameters:

```
source_service_instance_network_address: NA_i
source_plug_port: m
```

- Instance i of A gets to know the value of the parameter

```
dest_socket_new_port: l
```

Communication session establishment

- Acknowledgment of the established session is sent by the instance i of A to its agent.

```
type: session_ack
message_id: n
sub_type: service_to_agent
status: [status code]
source_plug_port: m
dest_socket_new_port: l
```

Communication session establishment

and forwarded to Manager by the agent:

```
type: session_ack
message_id: n
sub_type: agent_to_Manager
status: [status code]
source_plug_port: m
dest_socket_new_port: l
```

- Note that `message_id` is n (determined by instance i of A), and is the same for all the above request, response and acknowledgment messages.

Communication session establishment

- The complete list of the parameters of the session is as follows.

```
source_service_name: A
source_service_instance_network_address: NA_i
source_service_instance_id: i
source_plug_name: P
source_plug_port: m
dest_service_name: B
dest_service_instance_network_address: NA_j
dest_service_instance_id: j
dest_socket_name: S
dest_socket_port: k
dest_socket_new_port: l
```

Communication session establishment

- The number k is the port number (assigned by Manager) to the socket S (of instance j of B) for listening to clients.
- New port l of the socket S is dynamically assigned by instance j of B solely for the communication session with P of instance i of A.

- The instance i of A knows the above parameters except

```
            dest_service_instance_id: j
```

- The instance j of B knows the above parameters except

```
source_service_instance_id: i
source_service_name: A
```

- Manager knows all parameters of the session.

Closing a communication session
(P,S) between running instance i of A and instance j of B

- Each of the instances can initialize session closing, like in TCP connection, according to its own business logic.
- If instance i does so, it informs instance j of B that does the same; and vice versa.
- It is a regular closing of the session.

- A session may be closed only by one part of the communication due to failure of the other part or a broken link making the communication between these two parts impossible.
- In any of these cases above a running instance sends a message to its agent informing that the session was closed.
- Then, the agent forwards it to Manager.

Session closing by instance i of A, or by instance j of B

- The message from instance i of A to its agent is as follows.

```
type: source_service_session_close_info
message_id: n
sub_type: source_service_to_agent
source_service_name: A
source_service_instance_network_address: NA_i
source_service_instance_id: i
source_plug_name: P
source_plug_port: m
dest_service_name: B
dest_service_instance_network_address: NA_j
dest_socket_name: S
dest_socket_port: k
dest_socket_new_port: l
```

- The instance i of A sends all (known to it) parameters of the session.

Session closing by instance i of A, or by instance j of B

- The agent forwards the info to Manager:

```
type: source_service_session_close_info
message_id: n
sub_type: agent_to_Manager
source_service_name: A
source_service_instance_network_address: NA_i
source_service_instance_id: i
source_plug_name: P
source_plug_port: m
dest_service_name: B
dest_service_instance_network_address: NA_j
dest_socket_name: S
dest_socket_port: k
dest_socket_new_port: l
```

- The value of the parameter `message_id: n` is the same for the both messages above, and is determined by instance i of A.
- Manager can determine the identifier j of the instance of B on the basis of the port numbers: m, k and l.

Session closing by instance i of A, or by instance j of B

- Session closing by instance j of B is similar.
- The message from instance j of B to its agent is as follows.

```
type: dest_service_session_close_info
message_id: o
sub_type: dest_service_to_agent
source_service_instance_network_address: NA_i
source_plug_name: P
source_plug_port: m
dest_service_name: B
dest_service_instance_network_address: NA_j
dest_service_instance_id: j
dest_socket_name: S
dest_socket_port: k
dest_socket_new_port: l
```

- The instance j of B sends all (known to it) parameters of the session.

Session closing by instance i of A, or by instance j of B

- The agent forwards the info to Manager.

```
type: dest_service_session_close_info
message_id: o
sub_type: agent_to_Manager
source_service_instance_network_address: NA_i
source_plug_name: P
source_plug_port: m
dest_service_name: B
dest_service_instance_network_address: NA_j
dest_service_instance_id: j
dest_socket_name: S
dest_socket_port: k
dest_socket_new_port: l
```

- The value of the parameter `message_id: o` is the same for both messages above and is determined by instance j of B.
- Manager can determine the identifier i of the instance of A on the basis of the port numbers: m, k and l.

Session closing on the request of Manager

- An initiation of a communication session for abstract connection (A, (P,S), B) between instance i of A, and instance j of B is done by the instance i according to its business logic. Upon the request of instance i, configuration for such session is sent to instance i by Manager via the agent of instance i.

- By default, the socket S of instance j of B accepts the session establishment. This acceptance is known to Manager.

- Manager's request to close this session is only sent to the instance i of service A via the agent of instance i.
- The request contains all parameters of the session known to the instance i of A, i.e. except

```
dest_service_instance_id: j
```

Session closing on the request of Manager

- Request, from Manager to instance i of A via its agent to close a session, is as follows. The value o of the parameter `message_id:` is determined by Manager.

```
type: source_service_session_close_request
message_id: o
sub_type: Manager_to_agent
source_service_name: A
source_service_instance_network_address: NA_i
source_service_instance_id: i
source_plug_name: P
source_plug_port: m
dest_service_name: B
dest_service_instance_network_address: NA_j
dest_socket_name: S
dest_socket_port: k
dest_socket_new_port: l
```

Session closing on the request of Manager

- The agent forwards the request to instance i of A

```
type: source_service_session_close_request
message_id: o
sub_type: agent_to_source_service
source_service_name: A
source_service_instance_network_address: NA_i
source_service_instance_id: i
source_plug_name: P
source_plug_port: m
dest_service_name: B
dest_service_instance_network_address: NA_j
dest_socket_name: S
dest_socket_port: k
dest_socket_new_port: l
```

- Action of instance i of A: closing P.

Session closing on the request of Manager

- Response of instance i of A to its agent:

```
type: source_service_session_close_response
message_id: o
sub_type: source_service_to_agent
status: [status code]
```

- Forwarding the response to Manager.

```
type: source_service_session_close_response
message_id: o
sub_type: agent_to_Manager
status: [status code]
```

- After the successful session closing, the instance i may need a new communication session (for the same connection) to complete the task interrupted by the enforced closing. In order to do so the instance i can send a request to Manager for a configuration needed to establish such session. The message format of this request is given in a previous slide

- In the case of failure of instance i of service A, or its agent or node, a similar request must be sent to instance j of service B to close the session. This requires only minor modifications to the message sequence above.

Shutdown of service instances

- Graceful shutdown of a running instance of service by itself (on a request of Manager forwarded by agent) can be done after closing of all its communication sessions on the request of Manager via agent. The appropriate requests and responses are as follows.
- From Manager to agent:

```
type: graceful_shutdown_request
message_id: o
sub_type: Manager_to_agent
service_name: A
service_instance_id: i
```

- From agent to service instance:

```
type: graceful_shutdown_request
message_id: o
sub_type: agent_to_service_instance
service_name: A
service_instance_id: i
```

Shutdown of service instances

- Instance i of service A invokes internal method to shut down itself. Just before completing it, the instance sends the following response to the agent.

```
type: graceful_shutdown_response
message_id: o
sub_type: service_instance_to_agent
status: [status code]
```

- The agent forwards the response to Manager:

```
type: graceful_shutdown_response
message_id: o
sub_type: agent_to_Manager
status: [status code]
```

Shutdown of service instances

- Hard shutdown of instance i of service A is done by agent on the request of Manager.

```
type: hard_shutdown_request
message_id: n
sub_type: Manager_to_agent
service_name: A
service_instance_id: i
```

- Action of the agent: kill the process of service instance i.
- The response is as follows.

```
type: hard_shutdown_response
message_id: n
sub_type: agent_to_Manager
status: [status code]
```

Simple monitoring of service instances by agent

- Agent's request for observable metrics from a service instance is as follows.

```
type: health_control_request
message_id: o
sub_type: agent_to_service_instance
service_name: A
service_instance_id: i
```

Simple monitoring of service instances by agent

Service instance response:

```
type: health_control_response
message_id: o
sub_type: service_instance_to_agent
service_name: A
service_instance_id: i
status: [status code]
```

- The status codes are to express the metrics.
- Agent forwards the response to Manager only in the case of abnormal behavior (marked with a status code) of instance i.

```
type: health_control_response
message_id: o
sub_type: agent_to_Manager
service_name: A
service_instance_id: i
status: [status code]
```

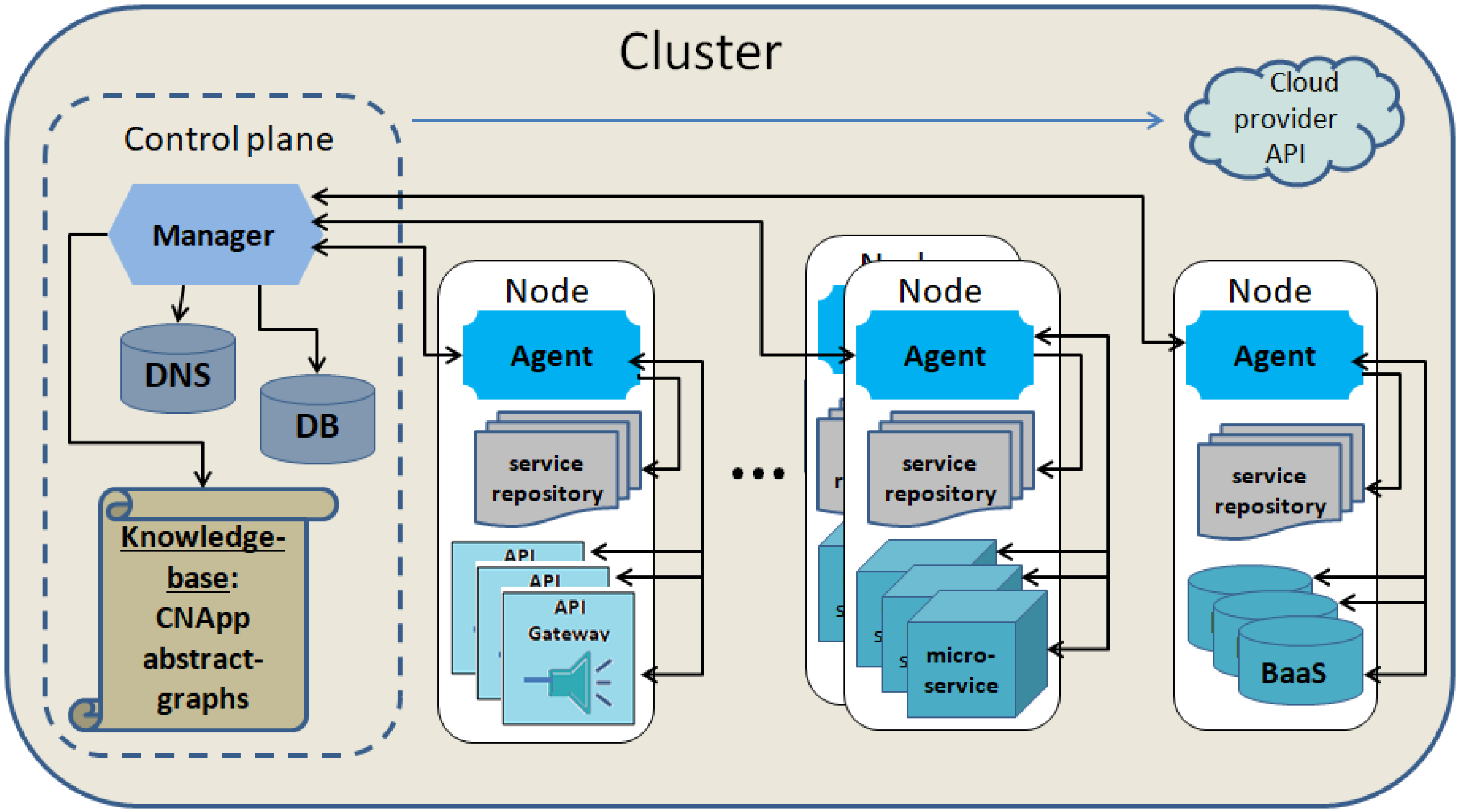
Final remarks

- Status codes can be used to handle failures. This is left to SSMMP implementations.
- Requirements for developing services of CNApp participating in SSMMP are as follows.
- Each instance of a service of CNApp is obliged to close its communication session at the request of the Manager. This can interfere with the business logic of the instance.
- For this reason, the current state of the communication session (until closed) must be stored in a BaaS service. To continue a task interrupted by the closing, the instance (at the client side of the connection) can establish a new session for the same abstract connection to continue and possibly to complete the task. Retrieval of the current state from the BaaS service may also be needed.

Final remarks

- This is the most complex requirement to be implemented in the codebase of each service participating in SSMMP.
- This requirement can also be seen as a standard recovery mechanism (independent of SSMMP) for handling failures of communication session, e.g. resulting from broken network connections. It seems reasonable to implement these recovery mechanisms in each CNApp service, regardless of SSMMP.
- The rest of the implementation requirements are relatively simple and can be completely separated from the business logic of the services.

The End


Cluster
Control plane
Manager
DNS
DB
Knowledge-base: CNApp abstract-graphs
Cloud provider API
Node
Agent
service repository
API
API
API Gateway
Node
Agent
service repository
micro-service
Node
Agent
service repository
BaaS
• • •