Workgroup: Network Working Group
Internet-Draft:
draft-ietf-anima-grasp-distribution-07
Published: 10 March 2023
Intended Status: Standards Track
Expires: 11 September 2023
Authors: S. Jiang (BUPT); A. Hecker (Huawei Technologies); B. Liu (Huawei Technologies); X. Xiao (Huawei Technologies); X. Zheng (Huawei Technologies); Y. Zhang (Individual)

Information Distribution over GRASP

Abstract

This document analyzes the Information distribution models in the Autonomic Networks that are based on the ANI. Most of instantaneous modes and their requirements have been met by GRASP already. However, in order to effectively support the asynchronous information distribution modes, which is newly described in this document, several new GRASP extensions are defined. This document also describes the corresponding behaviors on processing these new extensions.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

In Autonomic Networks [RFC7575], Autonomic Service Agents (ASAs) [RFC8993]running on autonomic nodes constantly exchange information, e.g. control/management signaling or data exchanging among ASAs. The

Autonomic Network Infrastructure (ANI) [RFC8993] provides generic support for these ASAs, mostly by GeneRic Autonomic Signaling Protocol (GRASP)[RFC8990]. This document introduces some important and typical use cases and analyzes their information distribution modes. Although most of instantaneous information distribution modes and their requirements have been met by GRASP already, asynchronous information distribution modes need new functions to support. In publishing for retrieval mode, information needs to be stored and re-distribute on-demand; additionally, conflict resolution is also needed when stored information is updated with information from multiple sources.

This document defines a series of GRASP extensions in order to support such information distribution mode. This document also describes the corresponding behaviors on processing them.

2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

This document uses terminology defined in [RFC7575].

3. Use Cases of Information Distribution

In this section, we present some important use cases where information distribution is required and Autonomic Control Plane (ACP) [RFC8994] support is commonly needed.

3.1. Service-Based Architecture (SBA) in 3GPP

In addition to Internet, carrier network (i.e. wireless mobile networks) is another world-wide networking system. The current architecture of 5G system defined by 3GPP follows a service-based architecture (SBA) where a network function (NF) can dynamically request a network service from another NF(s) when needed. Note that one NF can flexibly associate with multiple other NFs, instead of being physically wired to each other in a static way. NFs communicate with each other over service-based interface (SBI), which is also standardized by 3GPP [3GPP.23.501].

To realize an SBA network system, detailed requirements are further defined to specify how NFs should interact with each other with information exchange over the SBI in corresponding 3GPP technical specifications. We now list three services that are closely related to information distribution here.

1)

Service Exposure and Subscription: SBA requires that a NF can subscribe a particular service from another NF. This enables a NF to be notified when interested information occur. To achieve that, information (e.g., events, results, profiles, and statuses etc.) have to be stored first, and after that, whenever the information is requested, it has to be delivered properly to the requesting NF [TS23.502].

2) Network Repository Function (NRF): A particular network function where all service status information is stored for the whole network. An SBA network system requires all NFs to be stateless so as to improve the resilience as well as agility of providing network services. Therefore, the information of the available NFs and the service status generated by those NFs will be globally stored in NRF as a repository of the system. This clearly implies storage capability that keeps the information in the network and provides those information when needed. A concrete example is that whenever a new NF comes up, it firstly registers itself at NRF with its profile. When a network service requires a certain NF, it first inquires NRF to retrieve the availability information and decides whether there is an available NF, if not, a new NF must be instantiated [TS23.502].

3) Network Data Analysis Function (NWDAF): Data science technology is being quickly adopted in many industries, including 3GPP as well. It is a promising tool to retrieve valuable information from a large amount of data that is usually hard to be done by human being manually. NWDAF is a new NF added in to a 3GPP system. It is a NF dedicated to analyze the data collected from a network domain, which may consist of several sub-domains. Because of the importance of data-driven operations, distributed NWDAFs could exist in different domains in a 3GPP network, and among them, data storage and transferring will be needed because different sets of data are required for different tasks/purposes assigned to those NWDAFs in different domains. For example, if NWDAF uses machine learning techniques, the data required to train a neural network model will be different [TS23.501].

Notice that how the connectivity and trust among different NFs shall be bootstrapped and maintained by the control plane are not specified. In fact, 3GPP only considers the necessary requirements and features of a 3GPP network shall present. Hence, ACP and GRASP could be utilized as a specific solution and promoted to 3GPP.

3.2. In-Network Computing (INC)

In-network computing recently gets a lot of attentions [The-case-for-in-network-computing-on-demand]. INC improves the utilization of the computing resources in the network; INC also brings the processed results closer to the users, which may potentially improves the QoS of network services.

Unlike existing network systems, INC deploys computing tasks directly in the network rather than pushing the tasks to endpoints outside the network. Therefore, a network device is not just a transport device, but a mixture of forwarding, routing and computing. The requires an INC-supported network device having storage by default. Furthermore, computing agents deployed on network nodes will have to communicate with each other by exchanging information. There are several typical applications, where information distribution capability is required, which are summarized below.

1) Data Backup: There can be multiple computing agents that are created to serve the same purpose(s). In reality, the multiple agents can run for service resilience, load balancing and so on. This forms a service set. The instances in the service set can be deployed at different locations in the network while they need to keep synchronizing their local states for global consistency. In this case, the computing agents will have to constantly send and receive information across the network.

2) Data Aggregation: Multiple computing agents may process different computing tasks but the derived results have to be aggregated or combined. Then a collective result can be derived. In this case, different computing agents collaborate with each other, where information data are exchanged during the processing. A popular example is distributed AI or federated learning applications, where data are stored at different places and model training with the local data is also done in a distributed way. After that, trained models by distributed agents will have to be aggregated. Information distribution will be utilized heavily, combining with local storage.

Clearly, ASAs running on network nodes in ANI are the abstraction of the INC use case. ASAs can be deployed for both scenarios above.

3.3. Vehicle-to-Everything (V2X) Communications

The connected Autonomous Driving (AD) vehicles market is driving the evolution of the Internet of Vehicles (IoV) (or Vehicular IoT) and is growing at a five-year compound annual growth rate of 45%, which is 10 times faster than the overall car market. V2X communication is an inevitable enabling technology that connects vehicles to networks, where value-added services can be provided and enhance the functionalities of a vehicle. In this section, we introduce some use cases that will be closely relevant to information distribution in an ANI.

1) Real-time and High Definition Maps (HDM): In the era of autonomous driving, a digital map is not only for navigation, but real-time and detailed information is required when driving a vehicle. Real-time situational awareness is essential for autonomous vehicles especially at critical road segments in cases of changing road conditions (e.g. new traffic cone detected by another vehicle some time ago). In addition, the relevant high definition local maps have to be available with support from infrastructure side. In this regard, a digital map should not be considered static information stored on the vehicle, which is spontaneously updated in a periodical manner. Instead, it shall be considered a dynamic distribution based on information aggregated from the local area and such a distribution shall consider latency requirement. Clearly, the infrastructure side shall be able to hold the information in the network sufficiently close to the relevant area.

2) In-car Infotainment: This is another popular use case where in-car data demands will increase significantly in the near future. Today, users their mobile phone to access Internet for retrieving data for work or entertainment purposes. There is already a consensus among OTTs, carriers and car manufacturers that vehicle will become the center of information for passengers onboard. For entertainment, typical scenarios can be stereo HD video streaming and online gaming; for business purposes, examples can be mobile conference. This therefore requires the infrastructure side to be able to schedule and deliver requested information/data to the users with quality-of-service (QoS) considerations.

3) Software Update: Software components of connected cars will be remotely maintained in future. Therefore, software update has to be supported by the infrastructure side. Although this can be done by centralized solution where all vehicles access to a central clouds, in terms of load balancing and efficiency, prepared update components can be stored in the network and delivered to endpoints in a distributed manner.

Note that there could be different modes to support the potential use cases above. The first mode is that vehicles are not part of the ACP while simply accessing the edge nodes that are part of the ACP using information distribution to provide information required by the vehicles. The second mode is more radical where the vehicles also belong to the part of ACP while a dynamic ACP topology consisting of wireless link connectivity could exist. The latter scenario may further require all entities (both at the network side and the end point side) must be able to establish a trust layer relying on the security mechanism with Bootstrapping Remote Secure Key Infrastructure (BRSKI) [RFC8995].

3.4. Smart Home

Smart homes are designed to make home life much easier. Smart homes refer to a convenient home setup in which appliances and devices can be remotely controlled from anywhere using a mobile or other network device over an Internet connection. Devices in the smart home are connected over the Internet, allowing users to remotely control functions such as home security access, temperature, lighting, and a home theater. Smart home has considerable business prospects, and many Internet giants are investing in them, such as Amazon, Goolge and Apple. With the development of Internet technology, smart home user experience getting better and better. In this section, we present some use cases that are closely related to information distribution in an ANI.

1) Control Information: The control equipment often sends control information to specific devices in real time. For example, smart home with lighting control enables homeowners to reduce electricity use and benefit from energy-related cost savings. The control device sends an adjustment instruction to specific lights according to the ambient brightness in real-time.

2) Multi-Device Collaboration: Media and entertainment, which covers integrated entertainment systems in the home, including access and sharing of digital content on different devices, has proved to be the most prolific. Multi-device collaboration means that multiple devices work together to complete a service. In this case, distributed shared objects allow automatic synchronization of state or digital content between two or more devices. For example, users watch videos on tablets and/or TVs, and use their mobile phones to comment on and reply to the videos. In this way, concurrency, collaboration, and complementarity can be achieved. In this case, devices have to synchronize the information to the selected receivers. Compared with broadcast, sending information only to specific devices can save network traffic and improve network utilization.

4. Analysis of Information Distribution Modes and Requirements

According to the specific uses cases described in last section, this section summarizes the requirements of the use cases as a couple of general information distribution modes. Then in Section 4.3 , it described current gaps of GRASP protocol that could not fully support the distribution modes.

4.1. General Modes of Information Distribution

In a network (either in an Autonomic Network or any other networks), the way of distributing information could be modeled from the following two dimensions.

One dimension is from the perspective of the information distribution participants, there are two categories as below:

1) Point-to-point (P2P) Communication: information is exchanged between two nodes.

2) Point-to-Multi point (P2MP) Communication: information exchanges involve one source node and multiple receiving nodes.

The other dimension is from the timing perspective, also categoried as two modes as below:

1) Instantaneous mode: a source node sends the actual content (e.g. control/management signaling, synchronization data and so on to all interested receiver(s) immediately. Generally, some preconfigurations are required, where nodes interested in this information must be already known to all nodes because any source node must be able to decide, to which node the data is to be sent.

2) Asynchronous mode: here, a source node publishes the content in some forms in the network, which may later be looked for, found and retrieved by some other nodes. Here, depending on the size of the content, either the whole content or only its metadata might be published into the network. In the latter case the metadata (e.g. a content descriptor, e.g. a key, and a location in the network) may be used for the actual retrieval. Importantly, the source, i.e., here as a publisher, needs to be able to determine the location, where the information (or its metadata) can be stored.

Note that in both cases, the total size of transferred information can be larger than the payload size of a single message of a used transport protocol (e.g., Synchronization and Flood messages in GRASP). This document also gives support for bulk data transfer in Section 6.3.

4.2. ANI Requirements on Information Distribution

In ANI, on top of the general information distribution modes described in Section 4.1 , there are also ASA-level specific requirements of distributing information as the following:

1)

Long Communication Intervals. The actual sending of the information is not necessarily instantaneous with some events. Sophisticated ASAs may involve into longer jobs/tasks (e.g. database lookup, validations, etc.) when processing requests, and might not be able to reply immediately. Instead of actively waiting for the reply, a better way for an interested ASA might be to get notified, when the reply is finally available.

2) Common Interest Distribution. ASAs may share information that is a common interest. For example, the network intent [RFC9316] needs to be distributed to network nodes enrolled, which is usually P2MP mode. Intent distribution can also be performed by an instant flooding (e.g. via GRASP) to every network node. However, because of network changes, not every node can be just ready at the moment when the network intent is broadcast. Also, a flooding often does not cover all network nodes as there is usually a limitation on the hop number. In fact, nodes may join in the network sequentially. In this situation, an asynchronous communication mode could be a better choice where every (newly joining) node can subscribe the intent information and will get notified if it is ready (or updated).

3) Distributed Coordination. With computing and storage resources on autonomic nodes, alive ASAs not only consume but also generate data information. An example is ASAs coordinating with each other as distributed schedulers, responding to service requests and distributing tasks. It is critical for those ASAs to make correct decisions based on local information, which might be asymmetric as well. ASAs may also need synthetic/aggregated data information (e.g. statistic info, like average values of several ASAs, etc.) to make decisions. In these situations, ASAs will need an efficient way to form a global view of the network (e.g. about resource consumption, bandwidth and statistics). Obviously, purely relying on instant communication mode is inefficient, while a scalable, common, yet distributed data layer, on which ASAs can store and share information in an asynchronous way, should be a better choice.

4) Collision Update. Information data not only can be propagated and stored on network nodes in the network, they have to be conflict-free when information is updated especially when there is no central authority available. For example, when two ASAs try to propose different updates for the same piece of information that already exist in the network, a decision has to be made for how the existing information shall be updated. Obviously, if this duty has to be handled by individual ASAs, the implementation of an ASA is too complicated. Therefore, information distribution should consider conflict resolution and provides a set of general solutions for ASAs in order to keep information conflict free.

4.3. Gaps of current GRASP Protocol

As most of instantaneous information distribution modes and their requirements have been met by GRASP already, asynchronous information distribution modes need new functions to be supported. In publishing for retrieval mode, information needs to be stored and re-distribute on-demand; additionally, conflict resolution is also needed when stored information is updated with information from multiple sources.

To extend GRASP to support the ASA requirements, some extensions are defined in Section 5 .

5. New GRASP Extensions for the Conditional Information Distribution

5.1. Un-solicited Synchronization Message

In fragmentary CDDL, an Un-solicited Synchronization message follows the pattern:

```
unsolicited_synch-message = [M_UNSOLIDSYNCH, session-id,
objective]
```

A node SHOULD actively send a unicast Un-solicited Synchronization message with the Synchronization data, to another node. This SHOULD be sent to port GRASP_LISTEN_PORT at the destination address, which could be obtained by GRASP Discovery or other possible ways. The synchronization data are in the form of GRASP Option(s) for specific synchronization objective(s).

5.2. Selective-Flooding Option

Normal flooding mode has already been supported by GRASP. This section defines a new Selective-Flooding option. Since GRASP is based on CBOR (Concise Binary Object Representation) [RFC8949], the format of the Selective-Flooding option is described in the Concise Data Definition Language (CDDL) [RFC8610] as follows:

```
Selective-Flooding-option = [O_SELECTIVE_FLOOD, +O_MATCH-
CONDITION, match-object, action]
```

```
      O_MATCH-CONDITION = [O_MATCH-CONDITION, Obj1, match-rule,
      Obj2] Obj1 = text

      match-rule = GREATER / LESS / WITHIN / CONTAIN

      Obj2 = text

      match-object = NEIGHBOR / SELF

      action = FORWARD / DROP
```

The option field encapsulates a match-condition option which represents the conditions regarding to continue or discontinue flood the current message. For the match-condition option, the Obj1 and Obj2 are to objects that need to be compared. For example, the Obj1 could be the role of the device and Obj2 could be "RSG". The match rules between the two objects could be greater, less than, within, or contain. The match-object represents of which Obj1 belongs to, it could be the device itself or the neighbor(s) intended to be flooded. The action means, when the match rule applies, the current device just continues flood or discontinues.

5.3. Subscription Objective Option

In fragmentary CDDL, a Subscription Objective Option follows the pattern:

```
   objective = [Subscription, 2, 2, subobj]

   objective-name = Subscription

   objective-flags = 2

   loop-count = 2

   subobj = text
```

This option MAY be included in GRASP M_Synchronization, when included, it means this message is for a subscription to a specific object.

5.4. Unsubscription Objective Option

In fragmentary CDDL, a Unsubscription Objective Option follows the pattern:

```
   objective = [Unsubscription, 2, 2, unsubobj]

   objective-name = Unsubscription

   objective-flags = 2

   loop-count = 2

   unsubobj = text
```

This option MAY be included in GRASP M_Synchronization, when included, it means this message is for a un-subscription to a specific object.

5.5. Publishing Objective Option

In fragmentary CDDL, a Publishing Objective Option follows the pattern:

```
   objective = [Publishing, 2, 2, pubobj]

   objective-name = Publishing

   objective-flags = 2

   loop-count = 2

   pubobj = text
```

This option MAY be included in GRASP M_Synchronization, when included, it means this message is for active delivery of a specific object data.

6. Processing Behaviors on Autonomic Nodes

In this section, how a node should behave in order to support the two identified modes of information distribution is discussed. An ANI is a distributed system, so the information distribution module must be implemented in a distributed way as well.

6.1. Instant Information Distribution (IID) Sub-module

In this case, an information sender directly specifies the information receiver(s). The instant information distribution sub-module will be the main element.

6.1.1. Instant P2P Communication

IID sub-module performs instant information transmission for ASAs running in an ANI. In specific, IID sub-module will have to retrieve the address of the information receiver specified by an ASA, then deliver the information to the receiver. Such a delivery can be done either in a connectionless or a connection-oriented way.

Current GRASP provides the capability to support instant P2P synchronization for ASAs. A P2P synchronization is a use case of P2P information transmission. However, as mentioned in Section 3, there are some scenarios where one node needs to transmit some information to another node(s). This is different to synchronization because after transmitting the information, the local status of the information does not have to be the same as the information sent to the receiver. An extension to support instant P2P communication on GRASP is described in Section 5. A node SHOULD send a M_UNSOLIDSYNCH message to the GRASP_LISTEN_PORT of the corresponding node.

6.1.2. Instant Flooding Communication

IID sub-module finishes instant flooding for ASAs in an ANI. Instant flooding is for all ASAs in an ANI. An information sender has to specify a special destination address of the information and broadcast to all interfaces to its neighbors. When another IID sub-module receives such a broadcast, after checking its TTL, it further broadcast the message to the neighbors. In order to avoid flooding storms in an ANI, usually a TTL number is specified, so that after a pre-defined limit, the flooding message will not be further broadcast again.

In order to avoid unnecessary flooding, a selective flooding can be done where an information sender wants to send information to multiple receivers at once. An exemplary extension to support selective flooding on GRASP is described in Section 5.

When doing this, sending information needs to contain criteria to judge on which interfaces the distributed information should and should not be sent. Specifically, the criteria contain:

* O_MATCH- CONDITION in Selective-Flooding-option: matching condition, a set of matching rules such as addresses of recipients, node features and so on.

* action in Selective-Flooding-option: what the node needs to do when the Matching Condition is fulfilled. For example, the action could be forwarding or dropping the distributed message.

Sent information must be included in the message with Selective-Flooding-option distributed from the sender. The receiving node reacts by first checking the carried O_MATCH- CONDITION in the message to decide who should consume the message, which could be either the node itself, some neighbors or both. If the node itself is a recipient, action in Selective-Flooding-option is followed; if a neighbor is a recipient, the message is sent accordingly.

6.2. Asynchronous Information Distribution (AID) Sub-module

In asynchronous information distribution, sender(s) and receiver(s) are not immediately specified while they may appear in an asynchronous way. Firstly, AID sub-module enables that the information can be stored in the network; secondly, AID sub-module provides an information publication and subscription (Pub/Sub) mechanism for ASAs.

As sketched in the previous section, in general each node requires two modules: 1) Information Storage (IS) module and 2) Event Queue (EQ) module in the information distribution module. Details of the two modules are described in the following sections.

6.2.1. Information Storage

IS module handles how to save and retrieve information for ASAs across the network. The IS module uses a syntax to index information, generating the hash index value (e.g. a hash value) of the information and mapping the hash index to a certain node in ANI. Note that, this mechanism can use existing solutions. Specifically, storing information in an ANIMA network should be realized in the following steps.

1) ASA-to-IS Negotiation. An ASA calls the API provided by information distribution module (directly supported by IS sub-module) to request to store the information somewhere in the network. The IS module performs various checks of the request (e.g. permitted information size).

2) Storing Peer Mapping. The information block SHOULD be handled by the IS module in order to calculate/map to a peer node in the network. Since ANIMA network is a peer-to-peer network, a typical way is to use distributed hash table (DHT) to map information to a unique index identifier. For example, if the size of the information is reasonable, the information block itself can be hashed, otherwise, some meta-data of the information block can be used to generate the mapping.

3) Storing Peer Negotiation Request. Negotiation request of storing the information SHOULD be sent from the IS module to the IS module on the destination node. The negotiation request contains parameters about the information block from the source IS module. According to the parameters as well as the local available resource, the requested storing peer will send feedback the source IS module.

4) Storing Peer Negotiation Response. Negotiation response from the storing peer SHOULD be sent back to the source IS module. If the source IS module gets confirmation that the information can be stored, source IS module will prepare to transfer the information block; otherwise, a new storing peer must be discovered (i.e. going to step 7).

5) Information Block Transfer. Before sending the information block to the storing peer that already accepts the request, the IS module of the source node SHOULD check if the information block can be afforded by one GRASP message. If so, the information block MUST be directly sent by calling a GRASP API ([RFC8991]). Otherwise, a bulk data transmission is needed. It can utilize one of existing protocols that is independent of the GRASP stack. A session connectivity can be established to the storing peer, and over the connection the bulky data can be transmitted part by part. In this case, the IS module should support basic TCP-based session protocols such as HTTP(s) or native TCP.

6) Information Writing. Once the information block (or a smaller block) is received, the IS module of the storing peer SHOULD store the data block in the local storage.

7) (Optional) New Storing Peer Discovery. If the previously selected storing peer is not available to store the information block, the source IS module MUST identify a new destination node to start a new negotiation. In this case, the discovery can be done by using discovery GRASP API to identify a new candidate, or more complex mechanisms can be introduced.

Similarly, Getting information from an ANI should be realized in the following steps.

1) ASA-to-IS Request. An ASA accesses the IS module via the APIs exposed by the information distribution module. The key/index of the interested information SHOULD be sent to the IS module. An assumption here is that the key/index should be known to an ASA before an ASA can ask for the information. This relates to the publishing/subscribing of the information, which are handled by other modules (e.g. Event Queue with Pub/Sub supported by GRASP).

2) Storing Peer Mapping. IS module SHOULD map the key/index of the requested information to a peer that stores the information, and prepares the information request. The mapping here follows the same mechanism when the information is stored.

3) Retrieval Negotiation Request. The source IS module SHOULD send a request to the storing peer and asks if such an information object is available.

4) Retrieval Negotiation Response. The storing peer checks the key/ index of the information in the request, and replies to the source IS module. If the information is found and the information block can be afforded within one GRASP message, the information SHOULD be sent together with the response to the source IS module.

5) (Optional) New Destination Request. If the information is not found after the source IS module gets the response from the originally identified storing peer, the source IS module MUST discover the location of the requested information.

IS module can reuse distributed databases and key value stores like NoSQL, Cassandra, DHT technologies. Storage and retrieval of information are all event-driven responsible by the EQ module.

6.2.2. Event Queue

The Event Queue (EQ) module is to help ASAs to publish information to the network and subscribe/unsubscribe to interested information in asynchronous scenarios. Extensions to support information publishing, subscription and unsubscripiton on GRASP are described in Section 5. In an ANI, information generated on network nodes is an event labeled with an event ID, which is semantically related to the topic of the information. Key features of EQ module are summarized as follows.

1) Event Group: An EQ module provides isolated queues for different event groups. If two groups of ASAs could have completely different purposes, the EQ module allows to create multiple queues where only ASAs interested in the same topic will be aware of the corresponding event queue.

2) Event Prioritization: Events SHOULD have different priorities in ANI. This corresponds to how much important or urgent the event implies. Some of them are more urgent than regular ones. Prioritization allows ASAs to differentiate events (i.e. information) they publish, subscribe or unsubscribe to.

3) Event Matching: an information consumer has to be identified from the queue in order to deliver the information from the provider. Event matching keeps looking for the subscriptions in the queue to see if there is an exact published event there. Whenever a match is found, it will notify the upper layer to inform the corresponding ASAs who are the information provider and subscriber(s) respectively.

The EQ module on every network node operates as follows.

1) Event ID Generation: If information of an ASA is ready, an event ID SHOULD be generated according to the content of the information. This is also related to how the information is stored/saved by the IS module introduced before. Meanwhile, the type of the event SHOULD also be specified whether it is control plane data or user plane data.

2) Priority Specification: According to the type of the event, the ASA SHOULD specify its priority to say how this event is to be processed. By considering both aspects, the priority of the event will be determined.

3) Event Enqueue: Given the event ID, event group and its priority, a queue SHOULD be identified locally if all criteria can be satisfied. The event SHOULD be added into the queue, otherwise a new queue will be created to accommodate such an event.

4) Event Propagation: The published event SHOULD be propagated to the other network nodes in the ANIMA domain. A propagation algorithm SHOULD be employed to optimize the propagation efficiency of the updated event queue states.

5) Event Match and Notification: While propagating updated event states, EQ module in parallel SHOULD keep matching published events and its interested consumers. Once a match is found, the provider and subscriber(s) SHOULD be notified for final information retrieval.

The category of event priority is defined as the following. In general, there are two event types:

1) Network Control Event: This type of events are defined by the ANI for operational purposes on network control. A pre-defined priority levels for required system messages is suggested. For highest level to lowest level, the priority value ranges from NC_PRIOR_HIGH to NC_PRIOR_LOW as integer values. The NC_PRIOR_* values will be defined later according to the total number system events required by the ANI.

2) Custom ASA Event: This type of events are defined by the ASAs of users. This specifies the priority of the message within a group of ASAs, therefore it is only effective among ASAs that join the same message group. Within the message group, a group header/leader has to define a list of priority levels ranging from CUST_PRIOR_HIGH to CUST_PRIOR_LOW. Such a definition completely depends on the individual purposes of the message group. When a system message is delivered, its event type and event priority value have to be both specified.

Event contains the address where the information is stored, after a subscriber is notified, it directly retrieves the information from the given location.

6.3. Bulk Information Transfer

In both cases discussed previously, they are limited to distributing messages containing GRASP Objective Options that cannot exceed the GRASP maximum message size of 2048 bytes. This places a limit on the size of data that can be transferred directly in a GRASP message such as a Synchronization or Flood operation for instantaneous information distribution.

There are scenarios in autonomic networks where this restriction is a problem. One case is the distribution of network policy in lengthy formats such as YANG or JSON. Another case might be an Autonomic Service Agent (ASA) uploading a log file to the Network Operations Center (NOC). A third case might be a supervisory system downloading a software upgrade to an autonomic node. A related case might be installing the code of a new or updated ASA to a target node.

Naturally, an existing solution such as a secure file transfer protocol or secure HTTP might be used for this. Other management protocols such as syslog [RFC5424] or NETCONF [RFC6241] might also be used for related purposes, or might be mapped directly over GRASP. The present document, however, applies to any scenario where it is preferable to re-use the autonomic networking infrastructure itself to transfer a significant amount of data, rather than install and configure an additional mechanism.

The node behavior is to use the GRASP Negotiation process to transfer and acknowledge multiple blocks of data in successive negotiation steps, thereby overcoming the GRASP message size limitation. The emphasis is placed on simplicity rather than efficiency, high throughput, or advanced functionality. For example, if a transfer gets out of step or data packets are lost, the strategy is to abort the transfer and try again. In an enterprise network with low bit error rates, and with GRASP running over TCP, this is not considered a serious issue.

As for any GRASP operation, the two participants are considered to be Autonomic Service Agents (ASAs) and they communicate using a specific GRASP Objective Option, containing its own name, some flag bits, a loop count, and a value. In bulk transfer, we can model the ASA acting as the source of the transfer as a download server, and the destination as a download client. No changes or extensions are required to GRASP itself, but compared to a normal GRASP negotiation, the communication pattern is slightly asymmetric:

1)

The client first discovers the server by the GRASP discovery mechanism (M_DISCOVERY and M_RESPONSE messages).

2) The client then sends a GRASP negotiation request (M_REQ_NEG message). The value of the objective expresses the requested item (e.g., a file name - see the next section for a detailed example).

3) The server replies with a negotiation step (M_NEGOTIATE message). The value of the objective is the first section of the requested item (e.g., the first block of the requested file as a raw byte string).

4) The client replies with a negotiation step (M_NEGOTIATE message). The value of the objective is a simple acknowledgement (e.g., the text string 'ACK').

The last two steps SHOULD be repeated until the transfer is complete. The server SHOULD signal the end by transferring an empty byte string as the final value. In this case the client responds with a normal end to the negotiation (M_END message with an O_ACCEPT option).

Errors of any kind SHOULD be handled with the normal GRASP mechanisms, in particular by an M_END message with an O_DECLINE option in either direction. In this case the GRASP session terminates. It is then the client's choice whether to retry the operation from the start, as a new GRASP session, or to abandon the transfer. The block size must be chosen such that each step does not exceed the GRASP message size limit of 2048 bits.

7. Security Considerations

The distribution source authentication could be done at multiple layers:

* Outer layer authentication: the GRASP communication is within ACP ([RFC8994]). This is the default GRASP behavior.

* Inner layer authentication: the GRASP communication might not be within a protected channel, then there should be embedded protection in distribution information itself. Public key infrastructure might be involved in this case.

8. IANA Considerations

This document defines a new GRASP message named "M_UNSOLIDSYNCH" and a new option named "O_SELECTIVE_FLOOD" which need to be added to the "GRASP Messages and Options" registry defined by [RFC8990]. And this document defines three new GRASP Objectives, "Subscription", "Unsubscription" and "Publishing" which need to be added to the "GRASP Objective Names" .

9. Acknowledgements

Valuable comments were received from Zoran Despotovic, Brian Carpenter, Michael Richardson, Roland Bless, Mohamed Boucadair, Diego Lopez, Toerless Eckert and other participants in the ANIMA working group.

This document was produced using the xml2rfc tool [RFC7991].

10. Contributors

Brian Carpenter
School of Computer Science
University of Auckland
PB 92019
Auckland 1142
New Zealand

Email: brian.e.carpenter@gmail.com

11. References

11.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC5424] Gerhards, R., "The Syslog Protocol", RFC 5424, DOI 10.17487/RFC5424, March 2009, <https://www.rfc-editor.org/info/rfc5424>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8610] Birkholz, H., Vigano, C., and C. Bormann, "Concise Data Definition Language (CDDL): A Notational Convention to Express Concise Binary Object Representation (CBOR) and

JSON Data Structures", RFC 8610, DOI 10.17487/RFC8610, June 2019, <https://www.rfc-editor.org/info/rfc8610>.

[RFC8949] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", STD 94, RFC 8949, DOI 10.17487/RFC8949, December 2020, <https://www.rfc-editor.org/info/rfc8949>.

[RFC8990] Bormann, C., Carpenter, B., Ed., and B. Liu, Ed., "GeneRic Autonomic Signaling Protocol (GRASP)", RFC 8990, DOI 10.17487/RFC8990, May 2021, <https://www.rfc-editor.org/info/rfc8990>.

[RFC8994] Eckert, T., Ed., Behringer, M., Ed., and S. Bjarnason, "An Autonomic Control Plane (ACP)", RFC 8994, DOI 10.17487/RFC8994, May 2021, <https://www.rfc-editor.org/info/rfc8994>.

11.2. Informative References

[RFC7575] Behringer, M., Pritikin, M., Bjarnason, S., Clemm, A., Carpenter, B., Jiang, S., and L. Ciavaglia, "Autonomic Networking: Definitions and Design Goals", RFC 7575, DOI 10.17487/RFC7575, June 2015, <https://www.rfc-editor.org/info/rfc7575>.

[RFC7991] Hoffman, P., "The "xml2rfc" Version 3 Vocabulary", RFC 7991, DOI 10.17487/RFC7991, December 2016, <https://www.rfc-editor.org/info/rfc7991>.

[RFC8991] Carpenter, B., Liu, B., Ed., Wang, W., and X. Gong, "GeneRic Autonomic Signaling Protocol Application Program Interface (GRASP API)", RFC 8991, DOI 10.17487/RFC8991, May 2021, <https://www.rfc-editor.org/info/rfc8991>.

[RFC8993] Behringer, M., Ed., Carpenter, B., Eckert, T., Ciavaglia, L., and J. Nobre, "A Reference Model for Autonomic Networking", RFC 8993, DOI 10.17487/RFC8993, May 2021, <https://www.rfc-editor.org/info/rfc8993>.

[RFC8995] Pritikin, M., Richardson, M., Eckert, T., Behringer, M., and K. Watsen, "Bootstrapping Remote Secure Key Infrastructure (BRSKI)", RFC 8995, DOI 10.17487/RFC8995, May 2021, <https://www.rfc-editor.org/info/rfc8995>.

[RFC9316] Li, C., Havel, O., Olariu, A., Martinez-Julia, P., Nobre, J., and D. Lopez, "Intent Classification", RFC 9316, DOI 10.17487/RFC9316, October 2022, <https://www.rfc-editor.org/info/rfc9316>.

[The-case-for-in-network-computing-on-demand]
Tokusashi, Y., "The case for in-network computing on demand", DOI 10.1109/RECONFIG.2018.8641696, February 2019, <https://ieeexplore.ieee.org/document/8641696>.

[TS23.501] 3GPP, "Technical Specification Group Services and System Aspects; System architecture for the 5G System (5GS); Stage 2 (Release 18)", December 2022.

[TS23.502] 3GPP, "Technical Specification Group Services and System Aspects; Procedures for the 5G System (5GS); Stage 2 (Release 18)", December 2022.

Appendix A. Asynchronous ID Integrated with GRASP APIs

Actions triggered to the information distribution module will eventually invoke underlying GRASP APIs. Moreover, EQ and IS modules are usually correlated. When an ASA publishes information, not only such an event is translated and sent to EQ module, but also the information is indexed and stored simultaneously. Similarly, when an ASA subscribes information, not only subscribing event is triggered and sent to EQ module, but also the information will be retrieved by IS module at the same time.

*Storing and publishing information: This action involves both IS and EQ modules where a node that can store the information will be discovered first and related event will e published to the network. For this, GRASP APIs discover(), synchronize() and flood() are combined to compose such a procedure. In specific, discover() call will specific its objective being to "store_data" and the return parameters could be either an ASA_locator who will accept to store the data, or an error code indicating that no one could afford such data; after that, synchronize() call will send the data to the specified ASA_locator and the data will be stored at that node, with return of processing results like store_data_ack; meanwhile, such a successful event (i.e. data is stored successfully) will be flooded via a flood() call to interesting parties (such a multicast group existed).

*Subscribing and getting information: This action involves both IS and EQ modules as well where a node that is interested in a topic will subscribe the topic by triggering EQ module and if the topic is ready IS module will retrieve the content of the topic (i.e. the data). GRASP APIs such as register_objective(), flood(), synchronize() are combined to compose the procedure. In specific, any subscription action received by EQ module will be translated to register_objective() call where the interested topic will be the parameter inside of the call; the registration will be (selectively) flooded to the network by an API call of flood()

with the option we extended in this draft; once a matched topic is found (because of the previous procedure), the node finding such a match will call API synchronize() to send the stored data to the subscriber.

Authors' Addresses

Sheng Jiang
Beijing University of Posts and Telecommunications
No. 10 Xitucheng Road
Haidian District
Beijing
100083
China

Email: shengjiang@bupt.edu.cn

Artur Hecker
Huawei Technologies
Munich Research Center
Huawei Technologies
Riesstr. 25
80992 Muenchen
Germany

Email: artur.hecker@huawei.com

Bing Liu
Huawei Technologies
Q5, Huawei Campus
No.156 Beiqing Road
Hai-Dian District, Beijing
100095
P.R. China

Email: leo.liubing@huawei.com

Xun Xiao
Huawei Technologies
Munich Research Center
Huawei Technologies
Riesstr. 25
80992 Muenchen
Germany

Email: xun.xiao@huawei.com

Xiuli Zheng
Huawei Technologies

Q27, Huawei Campus
No.156 Beiqing Road
Hai-Dian District, Beijing
100095
P.R. China

Email: zhengxiuli@huawei.com

Yanyan Zhang
Individual
Texas
United States of America

Email: linna.purple@gmail.com