ANIMA WG   B. Liu
INTERNET-DRAFT   Huawei Technologies
Intended Status: Standard Track   X. Xiao
Expires: June 14, 2020   A. Hecker
                                                  MRC, Huawei Technologies
                                                              S. Jiang
                                                    Huawei Technologies
                                                          Z. Despotovic
                                                  MRC, Huawei Technologies
                                                     December 12, 2019

Information Distribution in Autonomic Networking
draft-liu-anima-grasp-distribution-13

Abstract

   This document proposes a solution for information distribution in
   autonomic networks. Information distribution is categorized into two
   different modes: 1) instantaneous distribution; 2) publication for
   retrieval. In the former case, the information is sent, propagates
   and is disposed of after reception. In the latter case, information
   needs to be stored in the network.

   The capabilities to distribute information are basic and fundamental
   needs for an autonomous network (cf. ANI [I-D.ietf-anima-reference-
   model]). This document describes typical use cases of information
   distribution in ANI and requirements to ANI, such that rich
   information distribution can be natively supported. The document
   proposes extensions to the autonomic nodes and suggests an
   implementation based on GRASP extensions as a protocol on the wire.

Status of this Memo

   http://www.ietf.org/1id-abstracts.html

   The list of Internet-Draft Shadow Directories can be accessed at
   http://www.ietf.org/shadow.html

Table of Contents

1 Introduction

In an autonomic network, autonomic functions (AFs) running on autonomic nodes constantly exchange information, e.g. AF control/management signaling or AF data exchange. This document discusses the information distribution capability of such exchanges between AFs.

Depending on the number of participants, the information can be distributed in in the following scenarios:

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

   2) One-to-Many Communication: information exchanges involve an information source and multiple receivers.

The approaches to information distribution can be chiefly categorized into two basic modes:

   1) An instantaneous mode (push): a source sends the actual content (e.g. control/management signaling, synchronization data and so on) to all interested receiver(s) immediately. Generally, some preconfiguration is required, as nodes interested in this information must be already known to all nodes in the sense that any receiving node must be able to decide, to which nodes this data is to be sent.

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

To avoid repetitive implementations by each AF developer, this document opts for a common support for information distribution implemented as a basic ANI capability, therefore available to all AFs. In fact, GRASP already provides part of the capabilities.

Regardless, an AF may still define and implement its own information distribution capability. Such a capability may then be advertised using the common information distribution capability defined in this document. Overall, ANI nodes and AFs may decide, which of the

information distribution mechanisms they want to use for which type of information, according to their own preferences (e.g. semantic routing table, etc.)

This document first analyzes requirements for information distribution in autonomic networks (Section 3) and then discuss the relevant node behavior (Section 4). After that, the required GRASP extensions are formally introduced (Section 5).

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

3. **Requirements for Information Distribution in ANI**

   The question of information distribution in an autonomic network can
   be discussed through particular use cases or more generally.
   Depending on the situation it can be quite simple or might require
   more complex provisions.

   Indeed, in the simplest case, the information can be sent:
   1) at once (in one packet, in one flow),
   2) straightaway (send-and-forget),
   3) to all nodes.

   Presuming 1), 2) and 3) hold, information distribution in smaller or
   scarce topologies can be implemented using broadcast, i.e.
   unconstrained flooding. For reasons well-understood, this approach
   has its limits in larger and denser networks. In this case, a graph
   can be constructed such that it contains every node exactly once
   (e.g. a spanning tree), still allowing to distribute any information
   to all nodes straightaway. Multicast tree construction protocols
   could be used in this case. There are reasonable use cases for such
   scenarios, as presented in Appendix B.

   A more complex scenario arises, if only 1) and 2) hold, but the
   information only concerns a subset of nodes. Then, some kind of
   selection becomes required, to which nodes the given information
   should be distributed. Here, a further distinction is necessary;
   notably, if the selection of the target nodes is with respect to the
   nature or position of the node, or whether it is with respect to the
   information content. If the first, some knowledge about the node
   types, its topological position, etc (e.g. the routing information
   within ANI) can be used to distinguish nodes accordingly. For
   instance, edge nodes and forwarding nodes can be distinguished in
   this way. If the distribution scope is primarily to be defined by the
   information elements, then a registration / join / subscription or
   label distribution mechanism is unavoidable. This would be the case,
   for instance, if the AFs can be dynamically deployed on nodes, and
   the information is majorily destined to the AFs. Then, depending on
   the current AF deployment, the distribution scope must be adjusted as
   well.

   If only 1) holds, but the information content might be required again
   and again, or might not yet be fully available, then more complex
   mechanisms might be required to store the information within the
   network for later, for further redistribution, and for notification
   of interested nodes. Examples for this include distribution of
   reconfiguration information for different AF instances, which might
   not require an immediate action, but only an eventual update of the
   parameters. Also, in some situations, there could be a significant

delay between the occurrence of a new event and the full content availability (e.g. if the processing requires a lot of time).

Finally, none of the three might hold. Then, along with the subscription and notification, the actual content might be different from its metadata, i.e. some description of the content and, possibly, its location. The fetching can then be implemented in different, appropriate ways, if necessary as a complex transport session.

In essence, as flooding is usually not an option, and the interest of nodes for particular information elements can change over time, ANI should support autonomics also for the information distribution.

This calls for autonomic mechanisms in the ANI, allowing participating nodes to 1) advertise or publish 2) look for or subscribe to 3) store 4) fetch/retrieve 5) instantaneously push information elements.

In the following cases, situations depicting diverse information distribution needs are discussed.

   1) Long Communication Intervals. The actual sending of the information is not necessarily instantaneous with some event. Advanced AFs may involve into longer jobs/tasks (e.g. database lookup, authentication 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 AF might be to get notified, when the reply is finally available.

   2) Common Interest Distribution. AFs may share interest in common information. For example, the network intent will be distributed to network nodes enrolled, which is usually one-to-many scenario. Intent distribution can also be performed by an instant flooding (e.g. via GRASP) to every network node. However, because of network dynamics, 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 model 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 AFs not only consume but also generate data information. An example is AFs coordinating with each other as distributed schedulers, responding to service requests and

distributing tasks. It is critical for those AFs to make correct decisions based on local information, which might be asymmetric as well. AFs may also need synthetic/aggregated data information (e.g. statistic info, like average values of several AFs, etc.) to make decisions. In these situations, AFs 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 model is inefficient, while a scalable, common, yet distributed data layer, on which AFs can store and share information in an asynchronous way, should be a better choice.

Therefore, for ANI, in order to support various communication scenarios, an information distribution module is required, and both instantaneous and asynchronous communication models should be supported. Some real-world use cases are introduced in Appendix A.

4. Node Behaviors

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.

4.1 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.

 4.1.1 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 mention 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. This is not directly support by existing GRASP.

 4.1.2 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. 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 Matching Condition: a set of matching rules such as addresses
      of recipients, node features and so on.

      o Action: what the node needs to do when the Matching Condition
      is fulfilled. For example, the action could be forwarding or
      discarding the distributed message.

   Sent information must be included in the message distributed from the
   sender. The receiving node reacts by first checking the carried
   Matching 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 field is followed; if
   a neighbor is a recipient, the message is sent accordingly.

   An exemplary extension to support selective flooding on GRASP is
   described in Section 5.

4.2 **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.

 4.2.1 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 will 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 will 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 will 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 is 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 will check if the information block can
      be afforded by one GRASP message. If so, the information block
      will be directly sent by calling a GRASP API. Otherwise, a bulk
      data transmission is needed. For that, there are multiple ways to

do it.

The first option is to utilize one of existing protocols that is
independent of the GRASP stack. For example, 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.

The second option is to directly use GRASP itself for bulky data
transferring. [I-D.carpenter-anima-grasp-bulk-04].

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

7) (Optional) New Storing Peer Discovery. If the previously
selected storing peer is not available to store the information
block, the source IS module will have to 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 will 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 will 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 maps 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 sends 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
will 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 will have to 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.

4.2.2 Event Queue The Event Queue (EQ) module is to help ASAs to publish information to the network and subscribe to interested information in asynchronous scenarios. 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 AFs could have completely different purposes, the EQ module allows to create multiple queues where only AFs interested in the same topic will be aware of the corresponding event queue.

2) Event Prioritization: Events can 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 AFs to differentiate events (i.e. information) they publish or subscribe 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 is 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 is also specified where it can be of control purpose or user plane data.

2) Priority Specification: According to the type of the event, the ASA may 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 is identified locally if all criteria can be
satisfied. If there is such a queue, the event will be simply
added into the queue, otherwise a new queue will be created to
accommodate such an event.

4) Event Propagation: The published event will be propagated to
the other network nodes in the ANIMA domain. A propagation
algorithm can 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 keeps matching published events and
its interested consumers. Once a match is found, the provider and
subscriber(s) will 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.

4.3 Summary

In summary, the general requirements for the information distribution
module on each autonomic node are realized by two sub-modules
handling instant communications and asynchronous communications,
respectively. For instantaneous mode, node requirements are simple,

calling for support for additional signaling. With minimum efforts,
reusing the existing GRASP is possible.

For asynchronous mode, information distribution module uses new
primitives on the wire, and implements an event queue and an
information storage mechanism. An architectural consideration on ANI
with the information distribution module is briefly discussed in
Appendix B.

5. Extending GRASP for Information Distribution

5.1 Realizing Instant P2P Transmission

This could be a new message in GRASP. In fragmentary CDDL, an Un-
solicited Synchronization message follows the pattern:

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

A node MAY actively send a unicast Un-solicited Synchronization
message with the Synchronization data, to another node. This MAY be
sent to port GRASP_LISTEN_PORT at the destination address, which
might 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 Realizing Instant Selective Flooding

Since normal flooding is already supported by GRASP, this section
only defines the selective flooding extension.

In fragmentary CDDL, the selective flooding follows the pattern:

    selective-flood-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 Realizing Subscription as An Event

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

        subscription-objection-option = [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 Un_Subscription Objective Option

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

      Unsubscribe-objection-option = [UNSUBSCRIB, 2, 2, unsubobj]
      objective-name = SUBSCRIPTION
      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 Publish Objective Option follows the pattern:

```
    publish-objection-option = [PUBLISH, 2, 2, pubobj] objective-name
    = PUBLISH
    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 a publish of a specific object
data.
6. Security Considerations

The distribution source authentication could be done at multiple
layers:

   o Outer layer authentication: the GRASP communication is within
   ACP (Autonomic Control Plane,
   [I-D.ietf-anima-autonomic-control-plane]). This is the default
   GRASP behavior.

   o 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.

7. IANA Considerations

   TBD.

8.   References

8.1 Normative References

   [I-D.ietf-anima-grasp]
            Bormann, D., Carpenter, B., and B. Liu, "A Generic
            Autonomic Signaling Protocol (GRASP)", draft-ietf-
            animagrasp-15 (Standard Track), October 2017.

8.2 Informative References

   [RFC7575] Behringer, M., "Autonomic Networking: Definitions and
            Design Goals", RFC 7575, June 2015

   [I-D.ietf-anima-autonomic-control-plane]
            Eckert, T., Behringer, M., and S. Bjarnason, "An Autonomic
            Control Plane (ACP)", draft-behringer-anima-autonomic-
            control-plane-13, December 2017.

   [I-D.ietf-anima-stable-connectivity-10]

                Eckert, T., Behringer, M., "Using Autonomic Control Plane
                for Stable Connectivity of Network OAM", draft-ietf-anima-
                stable-connectivity-10, February 2018.

   [I-D.ietf-anima-reference-model]
                Behringer, M., Carpenter, B., Eckert, T., Ciavaglia, L.,
                Pierre P., Liu, B., Nobre, J., and J. Strassner, "A
                Reference Model for Autonomic Networking", draft-ietf-
                anima-reference-model-05, October 2017.

   [I-D.du-anima-an-intent]
                Du, Z., Jiang, S., Nobre, J., Ciavaglia, L., and M.
                Behringer, "ANIMA Intent Policy and Format", draft-
                duanima-an-intent-05 (work in progress), February 2017.

   [I-D.ietf-anima-grasp-api]
                Carpenter, B., Liu, B., Wang, W., and X. Gong, "Generic
                Autonomic Signaling Protocol Application Program Interface
                (GRASP API)", draft-ietf-anima-grasp-api-00 (work in
                progress), December 2017.

   [I-D.carpenter-anima-grasp-bulk-04]
                Carpenter, B., Jiang, S., Liu, B., "Transferring Bulk Data
                over the GeneRic Autonomic Signaling Protocol (GRASP)",
                draft-carpenter-anima-grasp-bulk-04 (work in progress),
                July 3, 2019

   [3GPP.29.500]
                3GPP, "Technical Realization of Service Based
                Architecture", 3GPP TS 29.500 15.1.0, 09 2018

   [3GPP.23.501]
                3GPP, "System Architecture for the 5G System", 3GPP TS
                23.501 15.2.0, 6 2018,
                <http://www.3gpp.org/ftp/Specs/html-info/23501.htm>.

   [3GPP.23.502]
                3GPP, "Procedures for the 5G System", 3GPP TS 23.502
                15.2.0, 6 2018, <http://www.3gpp.org/ftp/Specs/html-
                info/23502.htm>.

   [5GAA.use.cases]
                White Paper "Toward fully connected vehicles: Edge
                computing for advanced automotive communications", 5GAA
                <http://5gaa.org/news/toward-fully-connected-vehicles-
                edge-computing-for-advanced-automotive-communications/>

Authors' Addresses

   Bing Liu
   Huawei Technologies
   Q27, Huawei Campus

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

   Email: leo.liubing@huawei.com

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

   Email: jiangsheng@huawei.com

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

   Emails: xun.xiao@huawei.com

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

   Emails: artur.hecker@huawei.com

   Zoran Despotovic
   Munich Research Center
   Huawei technologies
   Riesstr. 25, 80992, Muenchen, Germany

   Emails: zoran.despotovic@huawei.com

Appendix A. Real-world Use Cases of Information Distribution

The requirement analysis in [Section 3](#) shows that generally information distribution should be better of as an infrastructure layer module, which provides to upper layer utilizations. In this section, we review some use cases from the real-world where an information distribution module with powerful functions do plays a critical role there.

A.1 Service-Based Architecture (SBA) in 3GPP 5G

In addition to Internet, the telecommunication network (i.e. carrier mobile wireless networks) is another world-wide networking system. The architecture of the 5G mobile networks from 3GPP has been defined to follow a service-based architecture (SBA) where any network function (NF) can be dynamically associated with any other NF(s) when needed to compose a network service. Note that one NF can simultaneously associate with multiple other NFs, instead of being physically wired as in the previous generations of mobile networks. NFs communicate with each other over service-based interface (SBI), which is also standardized by 3GPP [3GPP.23.501].

In order 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. We now list three requirements that are related to information distribution here.

   1) NF Pub/Sub: Any NF should be able to expose its service status to the network and any NF should be able to subscribe the service status of an NF and get notified if the status is available. A concrete example is that a session management function (SMF) can subscribe to the REGISTER notification from an access management function (AMF) if there is a new user equipment trying to access the mobile network [3GPP.23.502].

   2) Network Exposure Function (NEF): A particular network function that is required to manage the event exposure and distributions. Specifically, SBA requires such a functionality to register network events from the other NFs (e.g. AMF, SMF and so on), classify the events and properly handle event distributions accordingly in terms of different criteria (e.g. priorities) [3GPP.23.502].

   3) 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 first of all
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 or not there is an
available NF or a new NF must be instantiated [3GPP.23.502].

(Note: 3GPP CT adopted HTTP2.0/JSON to be the protocol communicating
between NFs, but autonomic networks can also load HTTP2.0 with in
ACP.)

A.2 Vehicle-to-Everything

Connected car is one of scenarios interested in automotive
manufacturers, carriers and vendors. 5G Automotive Alliance - an
industry collaboration organization defines many promising use cases
where services from car industry should be supported by the 5G mobile
network. Here we list two examples as follows [5GAA.use.cases].

1) Software/Firmware Update: Car manufacturers expect that the
software/firmware of their car products can be remotely
updated/upgraded via 5G network, instead of onsite visiting their
4S stores/dealers offline as nowadays. This requires the network
to provide a mechanism for vehicles to receive the latest software
updates during a certain period of time. In order to run such a
service for a car manufacturer, the network shall not be just like
a network pipe anymore. Instead, information data have to be
stored in the network, and delivered in a publishing/subscribing
fashion. For example, the latest release of a software will be
first distributed and stored at the access edges of the mobile
network, after that, the updates can be pushed by the car
manufacturer or pulled by the car owner as needed.

2) Real-time HD Maps: Autonomous driving clearly requires much
finer details of road maps. Finer details not only include the
details of just static road and streets, but also real-time
information on the road as well as the driving area for both local
urgent situations and intelligent driving scheduling. This asks
for situational awareness at critical road segments in cases of
changing road conditions. Clearly, a huge amount of traffic data
that are real-time collected will have to be stored and shared
across the network. This clearly requires the storage capability,
data synchronization and event notifications in urgent cases from
the network, which are still missing at the infrastructure layer.

A.3 Summary

Through the general analysis and the concrete examples from the real-world, we realize that the ways information are exchanged in the coming new scenarios are not just short and instant anymore. More advanced as well as diverse information distribution capabilities are required and should be generically supported from the infrastructure layer. Upper layer applications (e.g. ASAs in ANIMA) access and utilize such a unified mechanism for their own services.

Appendix B. Information Distribution Module in ANI

This section describes how the information distribution module fits into the ANI and what extensions of GRASP are required [I-D.ietf-anima-grasp].

```
                +-------------------+
                |      ASAs         |
                +-------------------+
                         ^
                         |
                         v
   +-------------Info-Dist. APIs--------------+
   | +---------------+ +-------------------+ |
   | | Instant Dist. | | Asynchronous Dist. | |
   | +---------------+ +-------------------+ |
    +----------------------------------------+
                         ^
                         |
                         v
              +---GRASP APIs----+
              |      ACP        |
              +-----------------+
```
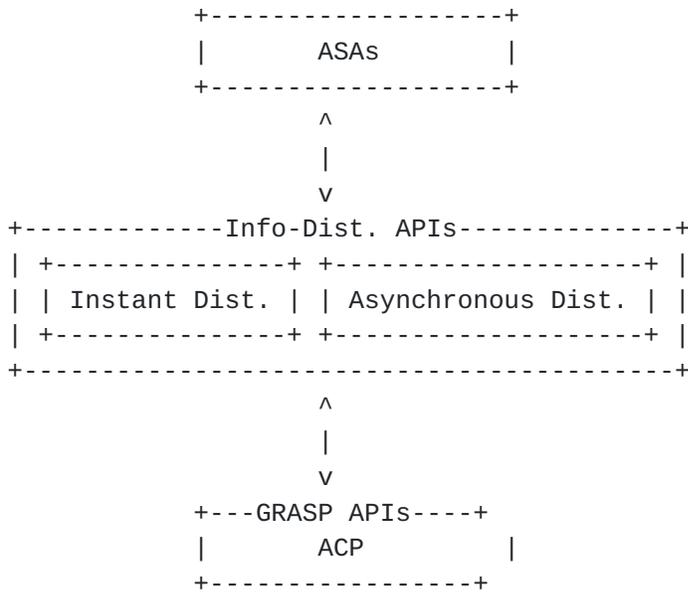
Figure 1. Information Distribution Module and GRASP Extension.

As the Fig 1 shows, the information distribution module two sub-modules for instant and asynchronous information distributions, respectively, and provides APIs to ASAs. Specific Behaviors of modules are described in Section 5.

Appendix C. 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 AF(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 AF(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.

o 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).

o 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.