# Event Service in Autonomic Networking

ANIMA WG, IETF 100, Singapore

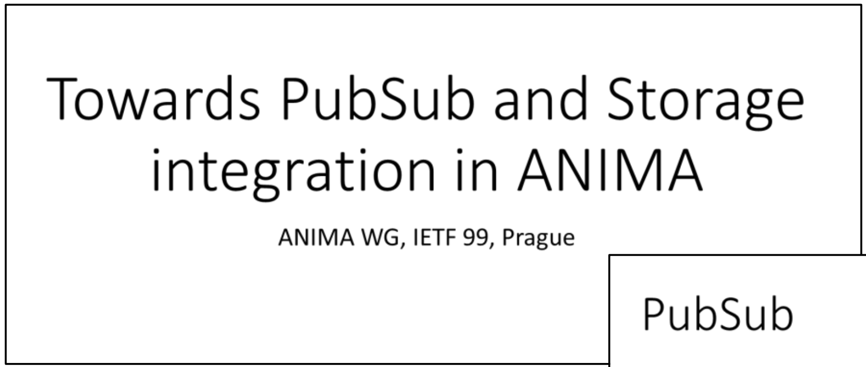
Xun Xiao, Artur Hecker, Zoran Despotovic

(https://datatracker.ietf.org/doc/draft-xiao-anima-event-service/)

## Call for new ideas

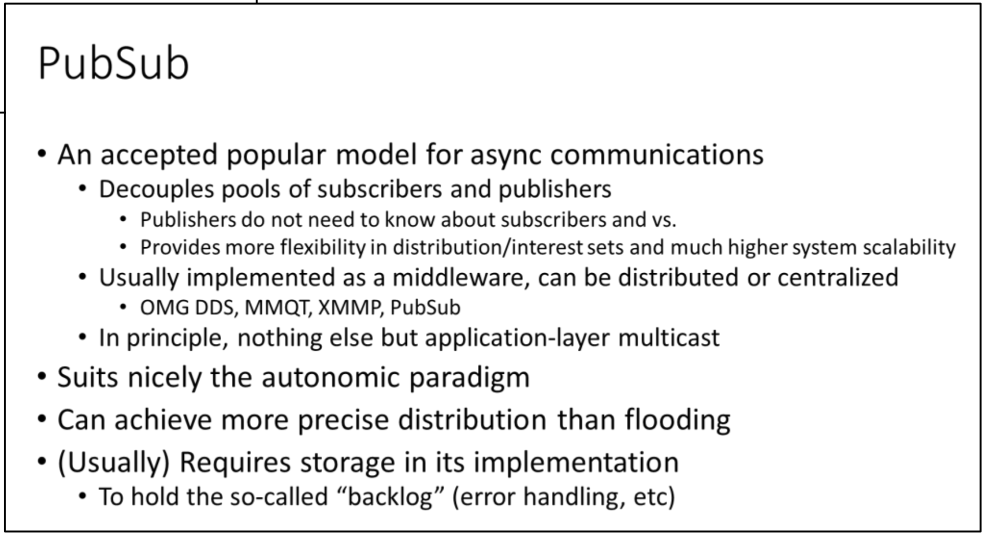
- Leveraging the Current ANI (GRASP, ACP and BRISK)
- **ANI extension & other reusable components for AN**
- Autonomic Service Agents over ANI and other reusable components

## Following what we proposed in IETF99'



Towards PubSub and Storage integration in ANIMA

ANIMA WG, IETF 99, Prague

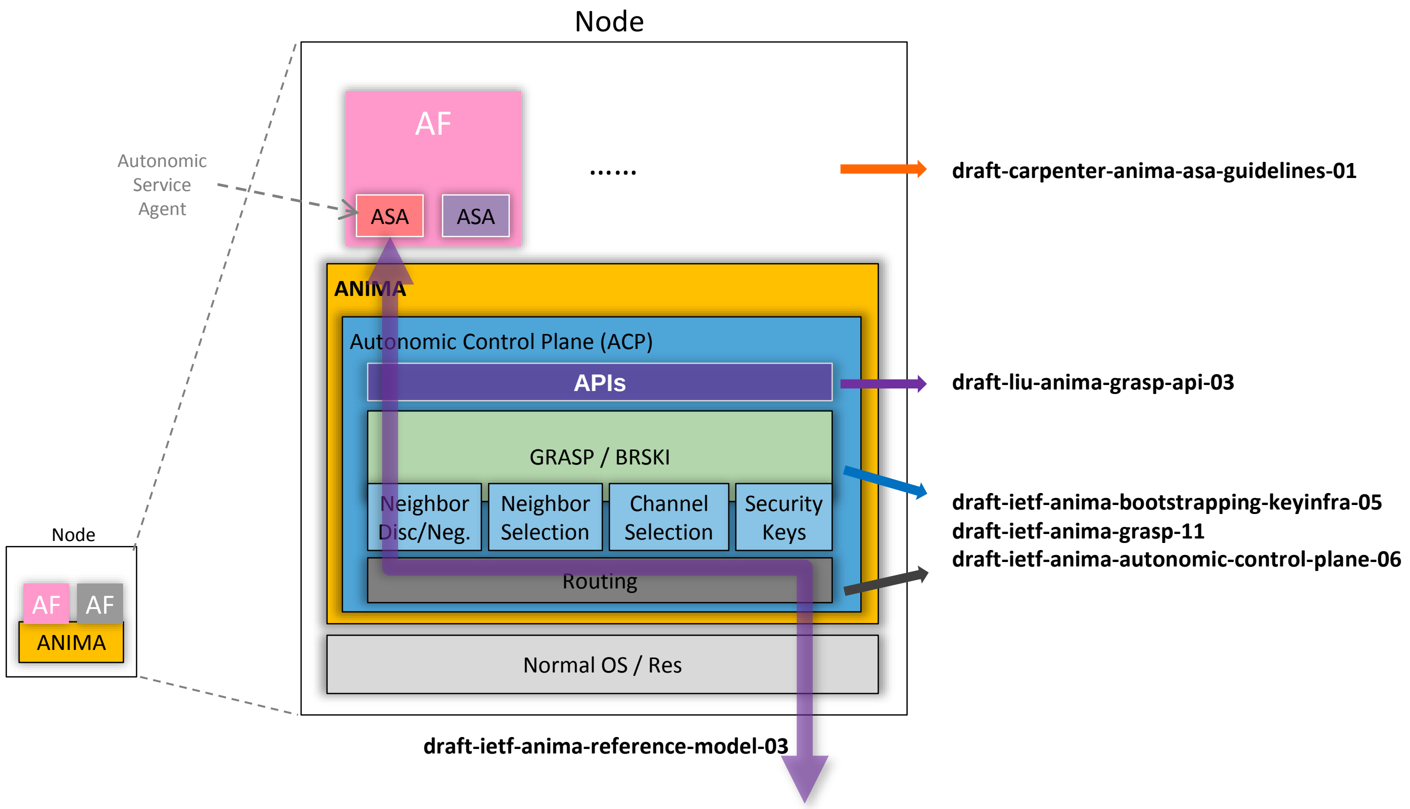


PubSub

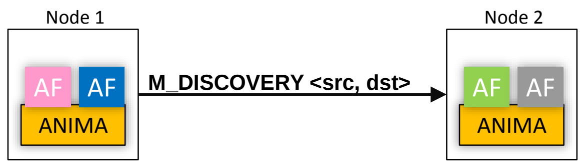
- An accepted popular model for async communications
  - Decouples pools of subscribers and publishers
    - Publishers do not need to know about subscribers and vs.
    - Provides more flexibility in distribution/interest sets and much higher system scalability
  - Usually implemented as a middleware, can be distributed or centralized
    - OMG DDS, MMQT, XMMP, PubSub
  - In principle, nothing else but application-layer multicast
- Suits nicely the autonomic paradigm
- Can achieve more precise distribution than flooding
- (Usually) Requires storage in its implementation
  - To hold the so-called "backlog" (error handling, etc)

- We propose to include event service (ES) into ANI as an extension and reusable component.

# Existing node architecture in ANIMA


Node
AF
ASA
ASA
......
Autonomic Service Agent
ANIMA
Autonomic Control Plane (ACP)
APIs
GRASP / BRSKI
Neighbor Disc/Neg.
Neighbor Selection
Channel Selection
Security Keys
Routing
Normal OS / Res
Node
AF
AF
ANIMA
draft-carpenter-anima-asa-guidelines-01
draft-liu-anima-grasp-api-03
draft-ietf-anima-bootstrapping-keyinfra-05
draft-ietf-anima-grasp-11
draft-ietf-anima-autonomic-control-plane-06
draft-ietf-anima-reference-model-03

# Communication models in existing ANI



- A tightly coupled client-service communication paradigm
  - A sender directly sends to a specified receiver
- Examples (from using GRASP):
  - Dynamic peer discovery (M_DISCOVERY, M_RESPONSE)
  - State synchronization (M_REQ_SYN, M_SYNCH, M_FLOOD)
  - Parameter settings negotiations (M_REQ_NEG, M_NEGOTIATE, M_WAIT, M_END)

- <u>Observations</u>:
  1. *NOT always known/available*.

     - Two interacting nodes (ASAs) are not always known to each other.

     - Two interacting nodes (ASAs) are not always available to each other.

     - e.g. churn, autonomic nodes may come and go in a dynamic way

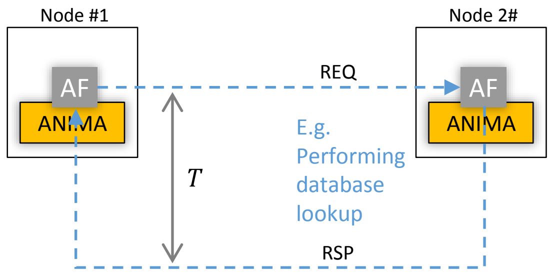
  2. *NOT always an instant reply*.

     - One node (an ASA) might be interested in some information, when certain criteria are satisfied / conditions are met.

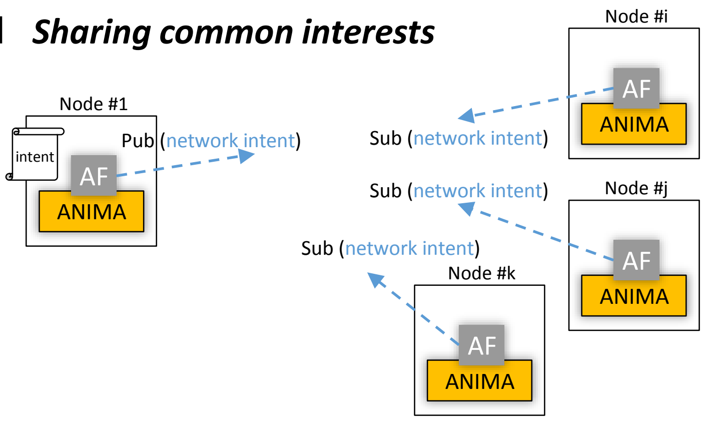
     - "When the average temperature is above x degrees, please let me know."
     - "When there is a new autonomic node joining in the ACP domain, please let me know."
     - …

# Use cases in ANIMA

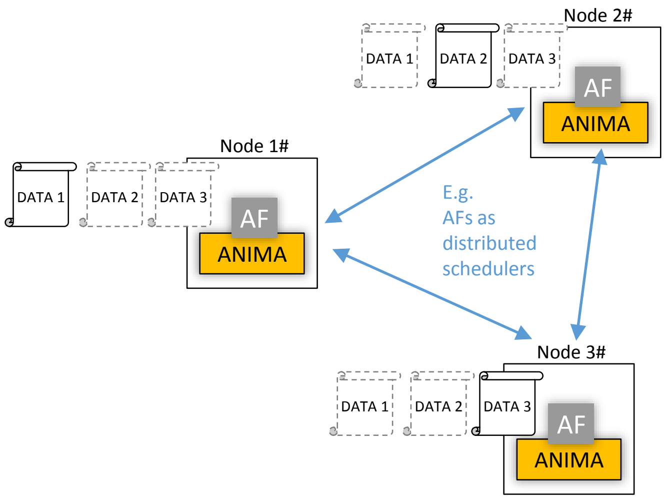
## ❑ *Replying taking long time*



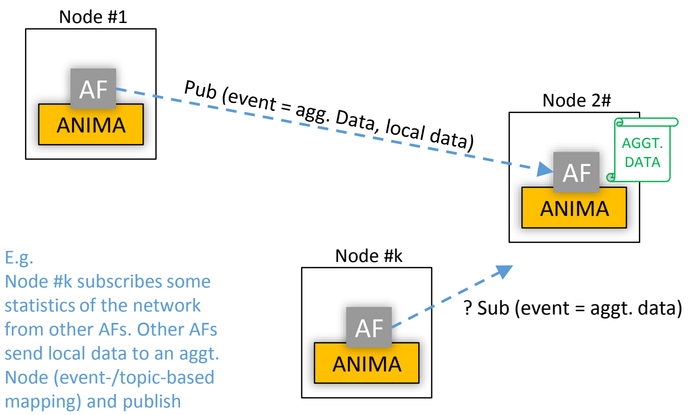
## ❑ *Sharing common interests*



## ❑ *Distributing data and build common views among AFs*

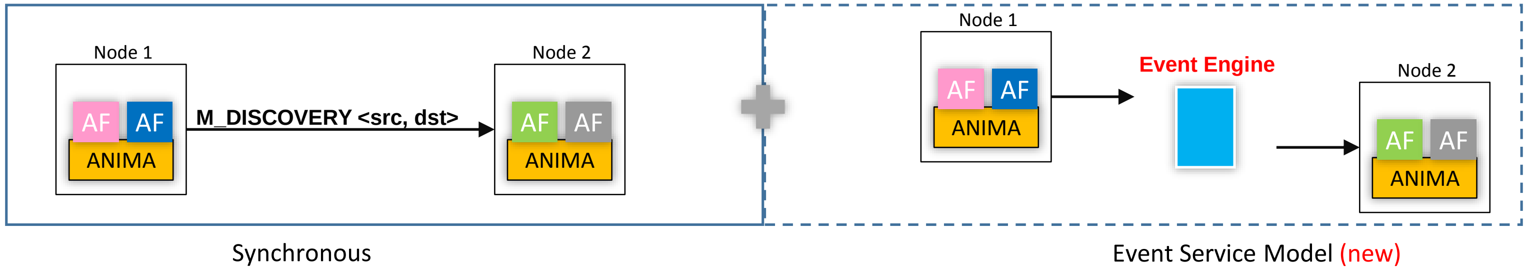


## ❑ *Distributing synthetic/aggregated data*


Node #1
AF
ANIMA
Pub (event = agg. Data, local data)
Node 2#
AGGT. DATA
AF
ANIMA
Node #k
AF
ANIMA
? Sub (event = aggt. data)

E.g. Node #k subscribes some statistics of the network from other AFs. Other AFs send local data to an aggt. Node (event-/topic-based mapping) and publish

## Proposal: Event Service for ANIMA Extension of comm. models in ANI



- In addition to the current synchronous communication model, we propose to extend ANI to support Event Service:
  - Two interacting parties are decoupled
  - Senders (Receivers) can publish (subscribe) information / request asynchronously
  - ANI is extended to be an information-driven system as well

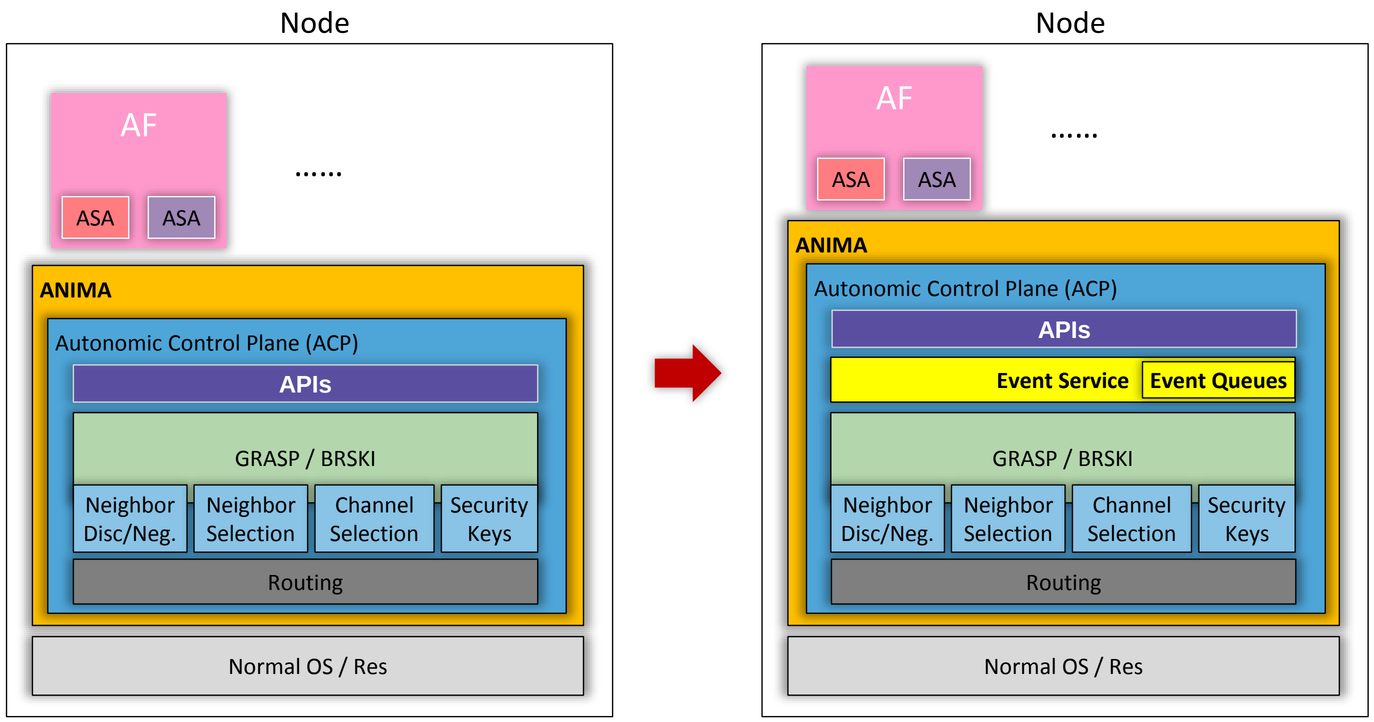
CONCENTRATE ON FEATURES / SERVICES HERE

## Integration alternatives

| | Standard-relevant | Integration Complexity | ANIMA conformance |
|---|---|---|---|
| 1. ES as a (must?) ASA | No *(as 3rd-party apps)* | *simple (handling events at the app layer)* | High *(Uses ANIMA w/o changes)* |
| **2. ES as an ANI extension** | Yes *(ACP will include the ES modules)* | *average (Adding new functions, but stand-alone)* | Average *(Follows model but extends it)* |

- Conclusion: we consider the 2nd option would be a proper way:
  - A clean design: will not introduce technical changes to existing ANI modules, an stand-alone extension;
  - Transparent to upper layers: only new interfaces will be added, don't care the technical details in ES;
  - Universal: every other modules follow the same interface / APIs

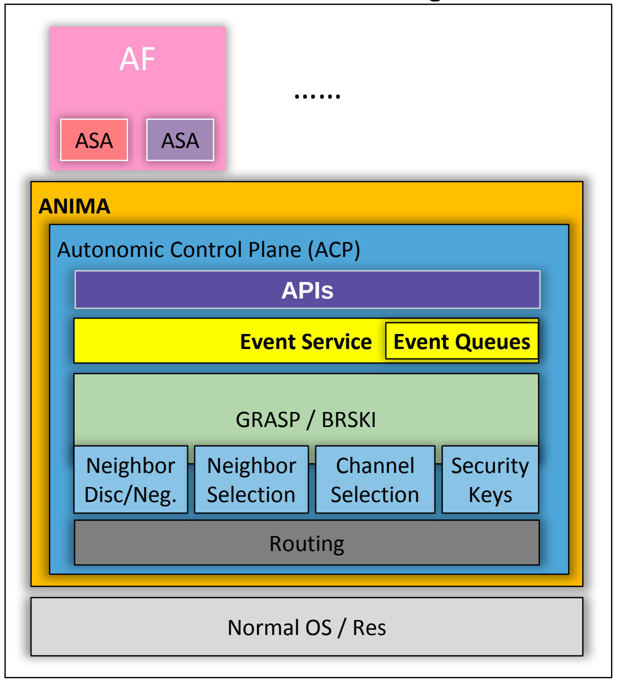
# Functionalities of the Event Service



- In the extended ANI, a new module is added on anima node, responsible for:
  - Listening to events from the ACP domain
  - Notifying other anima nodes when events occur
  - Handling local event subscription and publish
  - Storing information

# Impacted components

Node with an event engine



- ANIMA reference model:
  - A new module may have to be added in the reference model, functioning as an event engine middleware
- ANIMA GRASP:
  - New fields / functions may have to be added into GRASP so that it can support event-driven procedures, e.g.,

    - Fields: Event_id,

    - Messages: M_SUBSCRIBE, M_PUBLISH, M_NOTIFY, …
- Information distribution:
  - Can also call the extended APIs from GRASP for event-driven information distribution
- ANIMA GRASP – extended API library:
  - New APIs may have to be added so that enables upper layers ASAs (AFs) to have event-driven communications. For example:

    - Subscribe (Event_id, dst_id, … )

    - Publish (Event_id, dst_id, …)

## References

[1] draft-ietf-anima-reference-model-04
[2] draft-ietf-anima-grasp-15
[3] draft-liu-anima-grasp-api-05
[4] draft-liu-anima-grasp-distribution-04