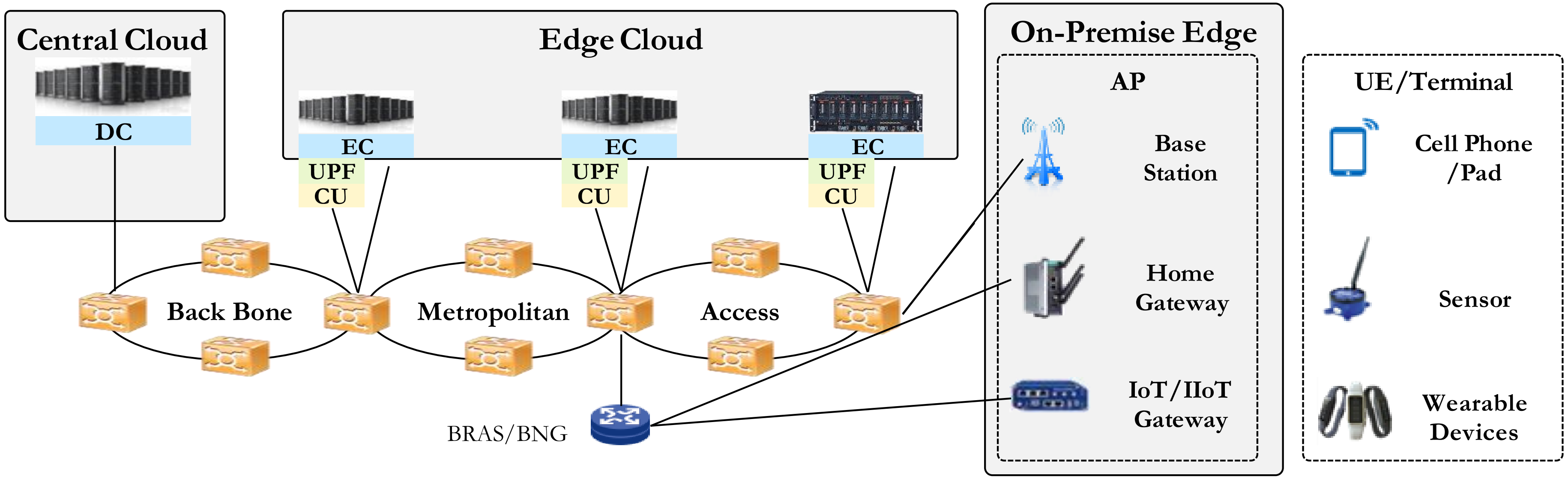
Dyncast in CFN

(dynamic anycast in compute first networking)

draft-geng-rtgwg-cfn-dyncast-ps-usecase
draft-li-rtgwg-cfn-dyncast-architecture

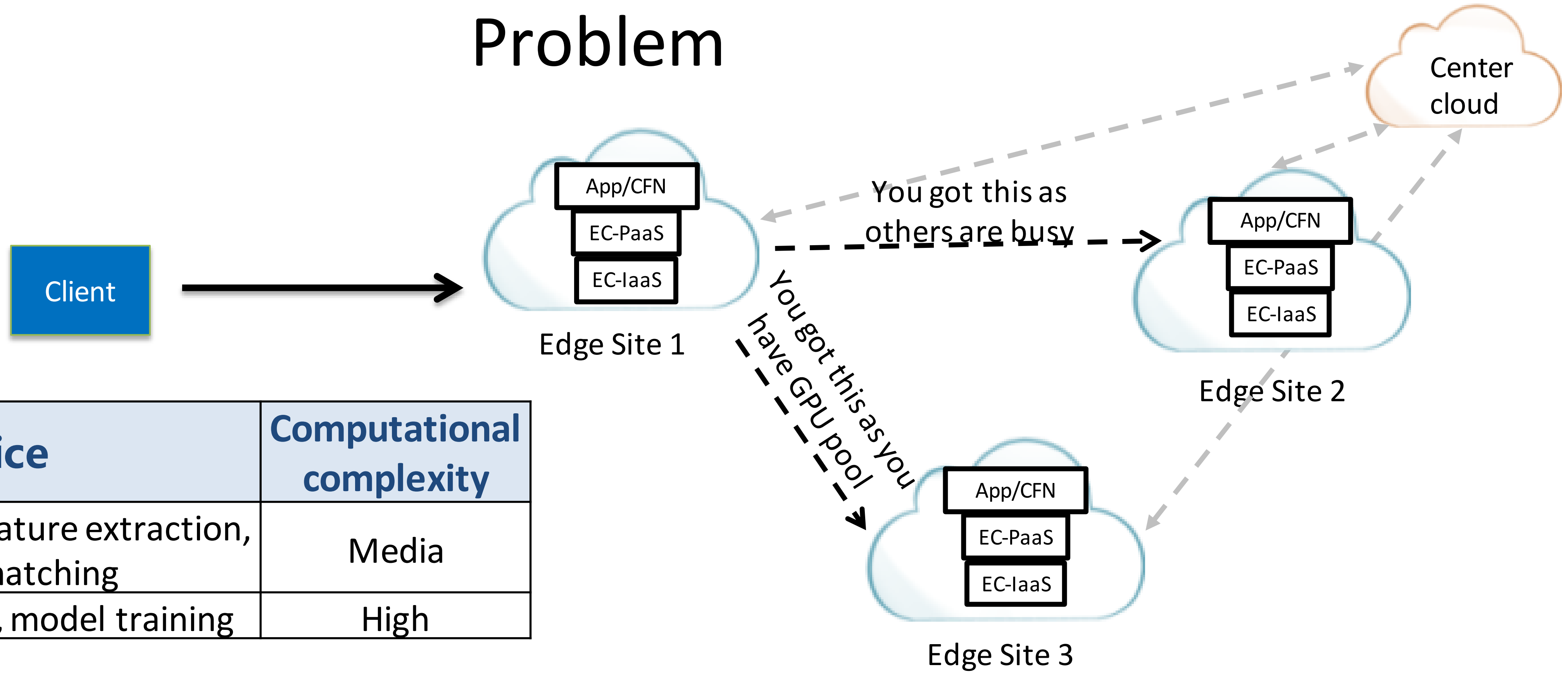
liyizhou@huawei.com

Typical Multi-edge Computing Usage Scenario



- Service providers are exploring the edge computing (AR, VR, connected Car)
  - Shorter, faster - compare with cloud
  - Energy (battery) saving, dataset size & governance consideration – compare with host side
- large number of edge sites in a city
- limited and varying computing resource for each site

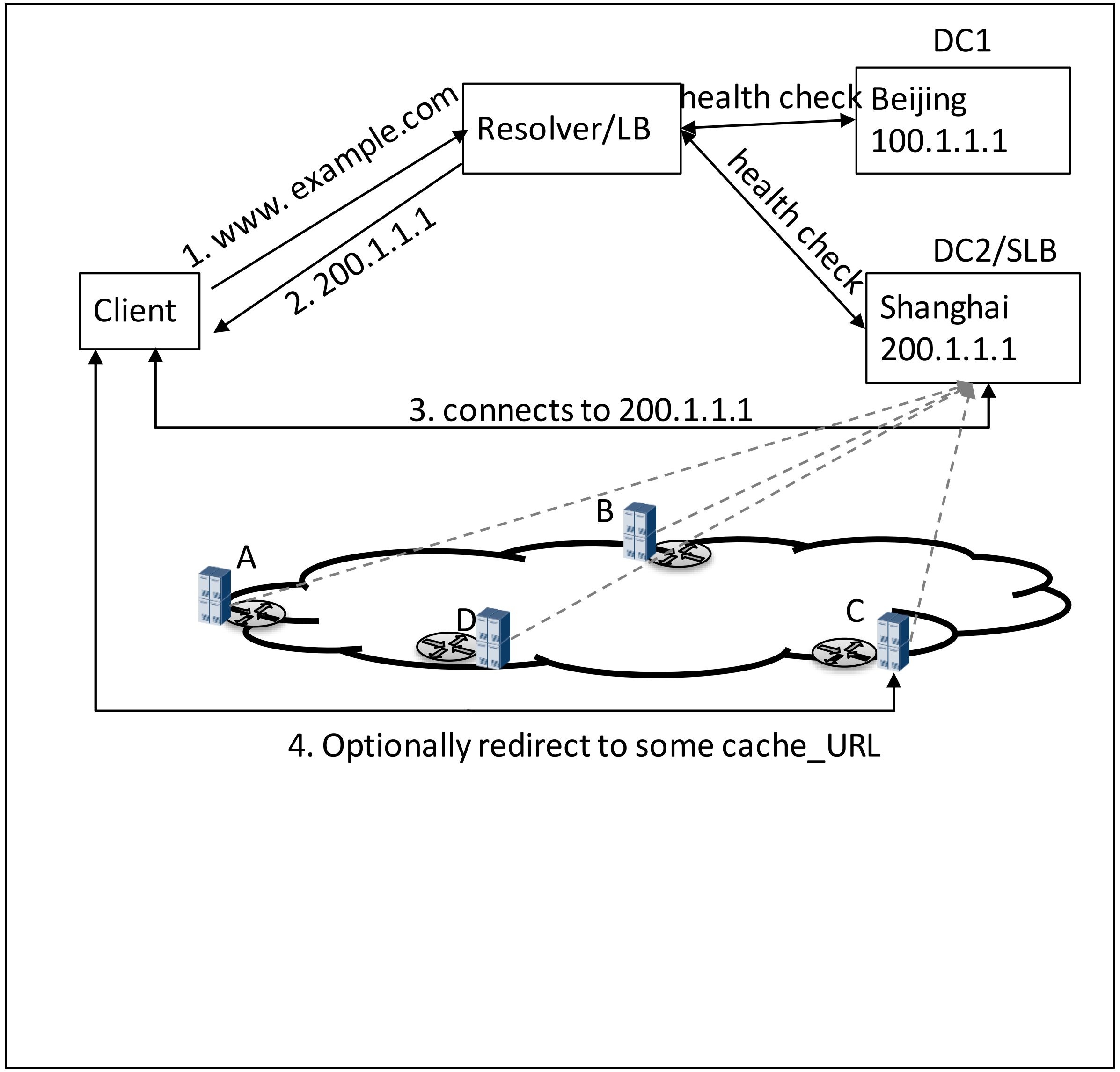
Problem



Service	Computational complexity
Object detection, feature extraction, template matching	Media
Object recognition, model training	High

Problem: How to optimally route service demands based on computing and network metrics to the best edge?

Current Practices, considerations and gaps – efficiency and latency

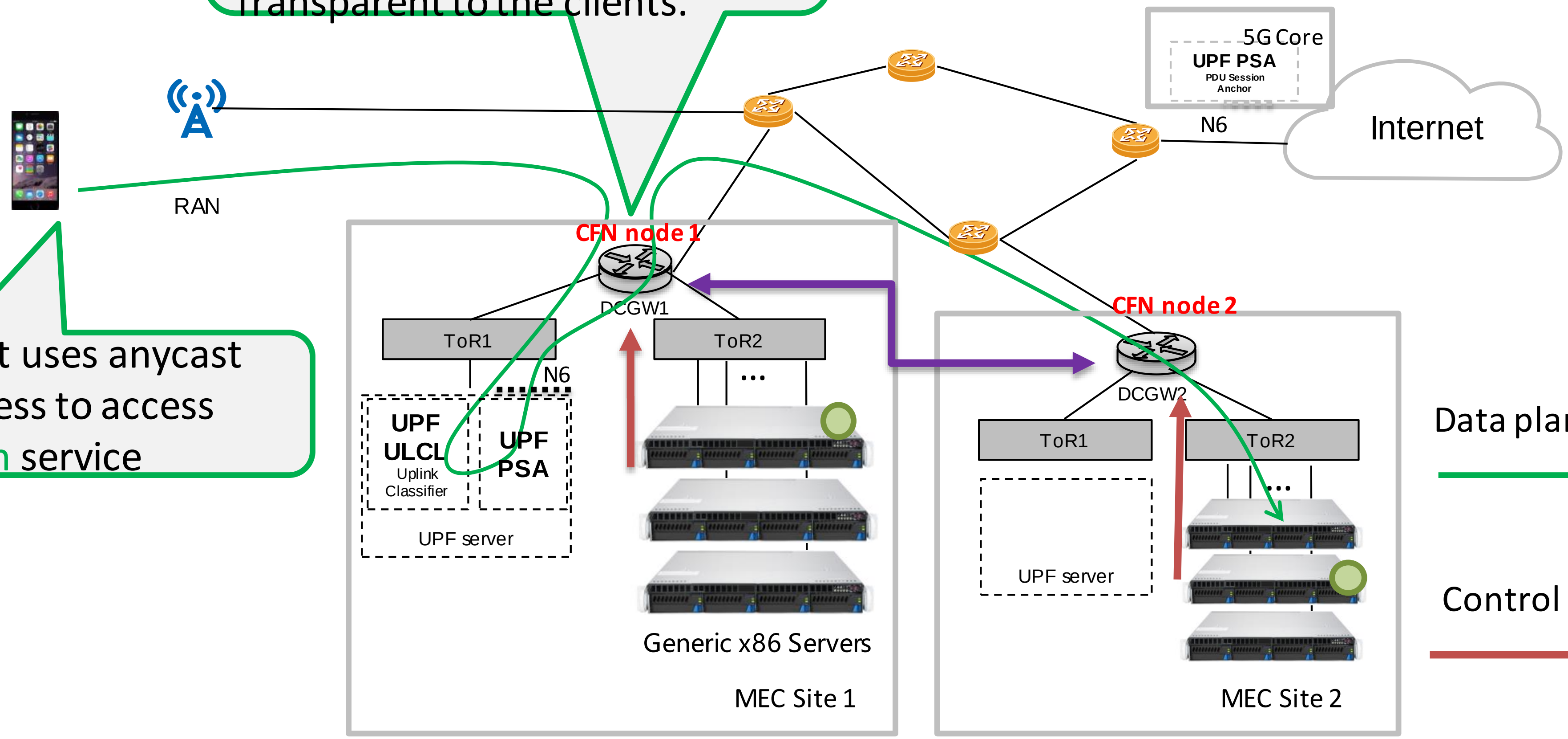


- Use geographical location, pick closest
  - Edges are not so far apart. Locations do not matter most.
- Health check in an infrequent base (>1s), switch when fail-over
  - Limited computing resources on edge, change rapidly (<1s)
- Random or round robin pick, network cost is not a concern or updated infrequently just to keepalive
  - Edges are not deployed in equal cost way, network status is considered at a later stage not at the same time
- Centralized determination, good for content retrieval.
  - Not be as good as for computation which has more dynamic nature and larger number
- Early binding: clients query first and then steer traffic.
  - Edge computing flow can be short. Early binding has high overhead.
- Caching at the client.
  - Stale info could be used, hard to guarantee to route optimally for each service demand
- Others:
  - Network based solution uses least network cost, computing load is hardly considered
  - Traditional anycast bases on single request/reply packet, no flow affinity

Concept of CFN-Dyncast

Routing the packets to the best edge with joint consideration of computing and network load. Transparent to the clients.

Client uses anycast address to access green service



Data plane:

🟩 Data flow to the selected MEC Site 2

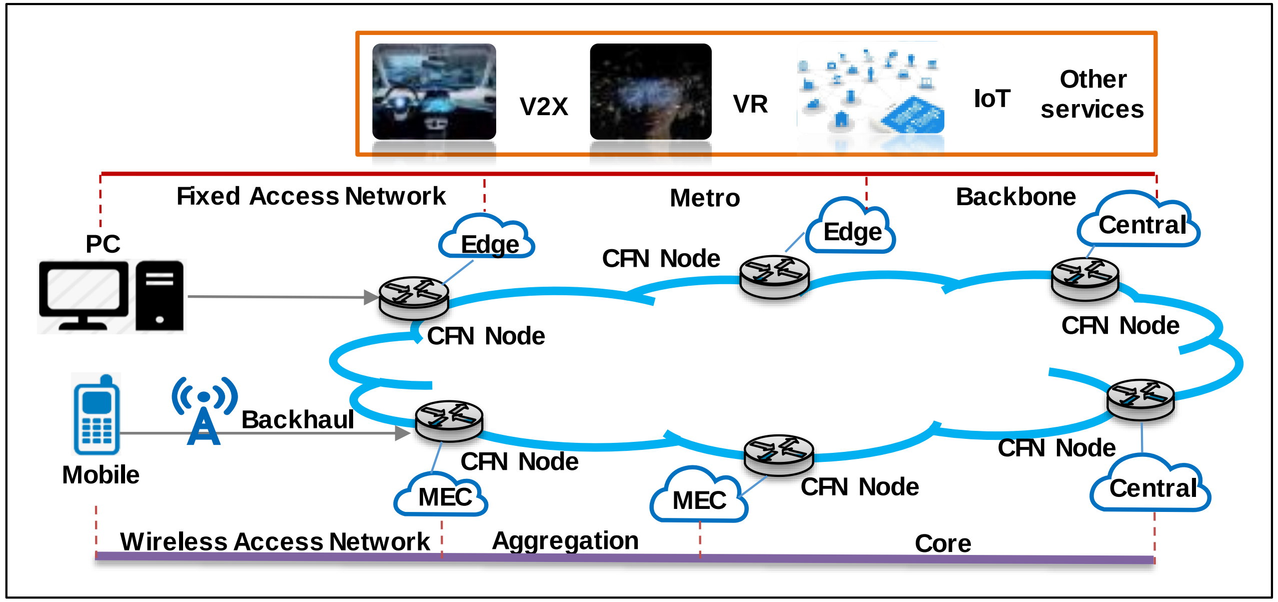
Control plane:

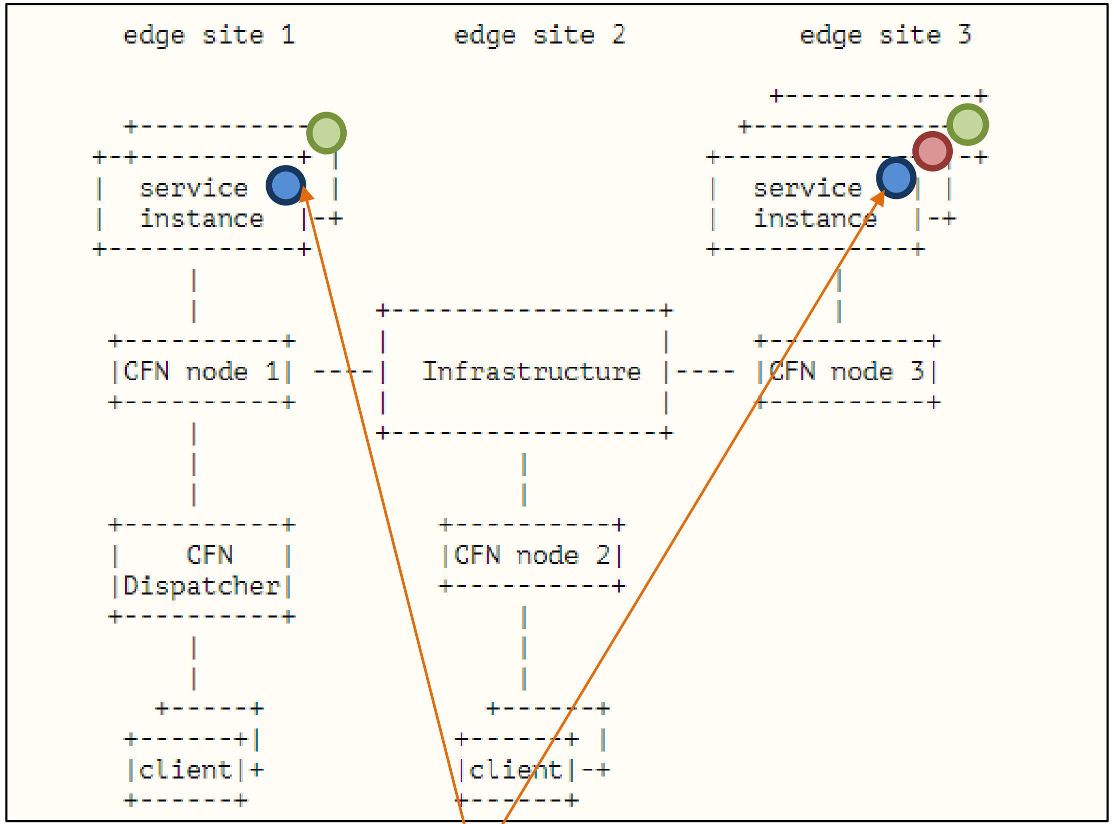
🟥 Service info notification and update to CFN node

🟪 Info distribution among CFN nodes

CFN: Compute first networking. A distributed computing environment across a network to perform flexible load management and performance optimizations.

Service and Service Instance in CFN-Dyncast





```
     edge site 1              edge site 2              edge site 3

                                                    +------------+
      +----------                                 +-----------
   +-+------------+                             +-------------  -+
   |  service      |                            |  service    |  |
   |  instance    |-+                           |  instance   |-+
   +--------------+                             +-------------+
          |                                            |
          |               +-----------------+          |
   +----------+           |                 |      +----------+
   |CFN node 1| ----|     Infrastructure    |----  |CFN node 3|
   +----------+           |                 |      +----------+
          |               +-----------------+
          |                        |
          |                        |
   +----------+              +----------+
   |   CFN    |              |CFN node 2|
   |Dispatcher|              +----------+
   +----------+                    |
          |                        |
          |                        |
      +-----+                  +-----+
   +------+|                +------+ |
   |client|+                |client|-+
   +------+                 +------+
```

Services:
- ● Blue
- ● Red
- ● Green

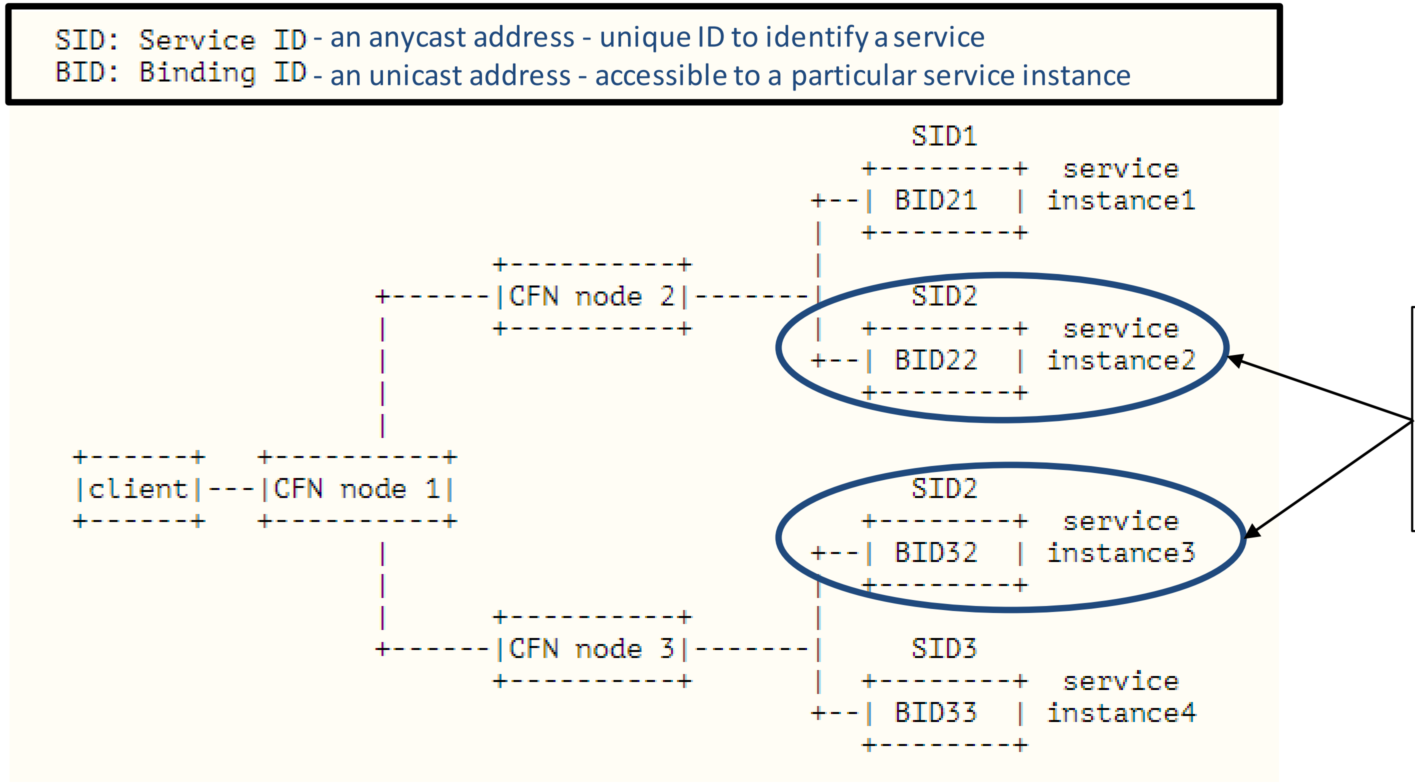
- A service offer one specific function no matter where it is deployed

- One Service can have several instances running on different edges
- Service instance is a running environment (e.g., a node) that makes the functionality of a service available
- All service instances running the same service are identified by the same **Service Identifier (SID)**

Features to be supported

- Anycast based service addressing methodology
- Flow affinity
- Computing Aware Routing

Anycast based service addressing methodology

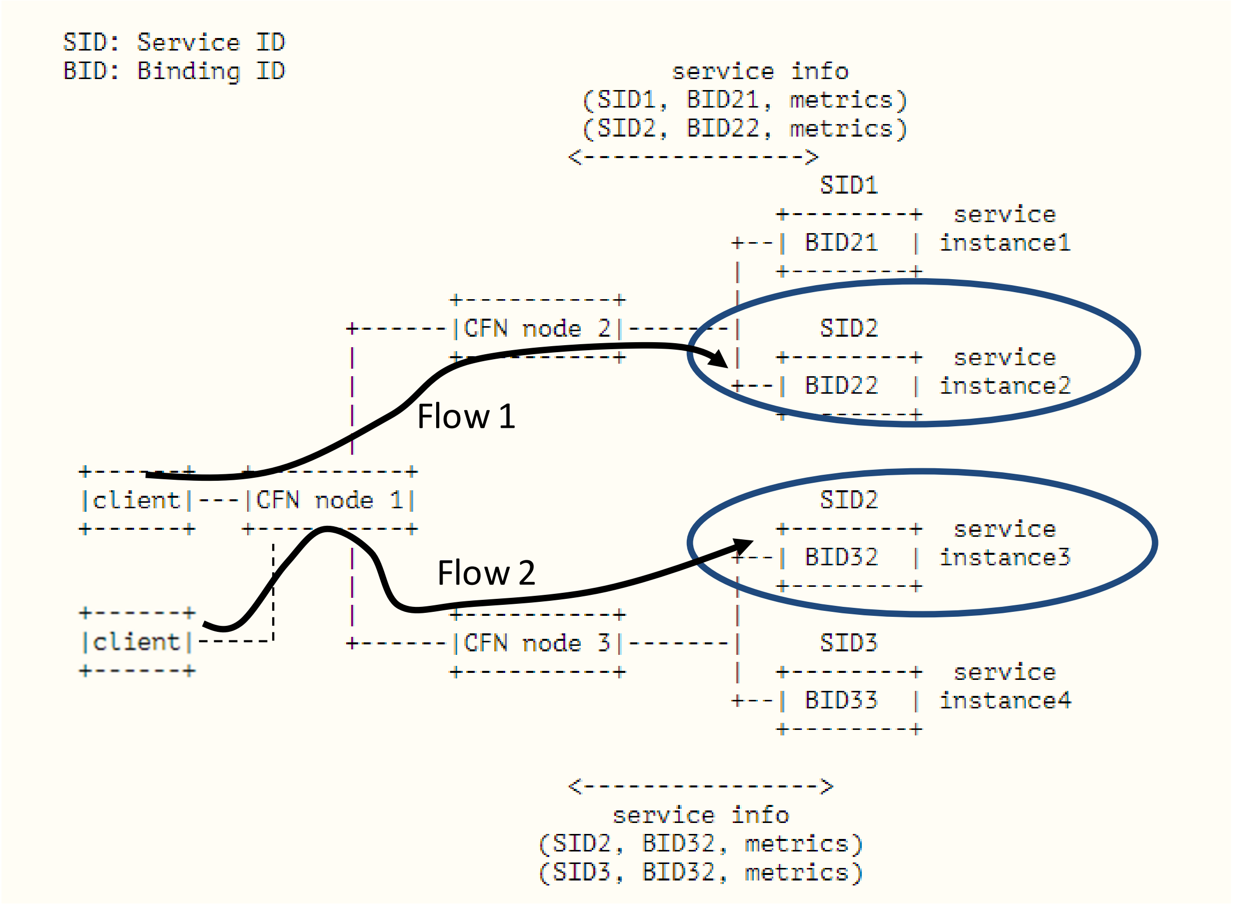


SID: Service ID - an anycast address - unique ID to identify a service
BID: Binding ID - an unicast address - accessible to a particular service instance

```
                                                    SID1
                                                 +--------+  service
                                              +--| BID21  | instance1
                                              |  +--------+
                          +-----------+       |
                   +------|CFN node 2|--------|     SID2
                   |      +-----------+       |  +--------+  service
                   |                          +--| BID22  | instance2
                   |                             +--------+
                   |
+------+   +-----------+
|client|---|CFN node 1|                             SID2
+------+   +-----------+                         +--------+  service
                   |                          +--| BID32  | instance3
                   |                          |  +--------+
                   |      +-----------+       |
                   +------|CFN node 3|--------|     SID3
                          +-----------+       |  +--------+  service
                                              +--| BID33  | instance4
                                                 +--------+
```

- Same service, i.e. same service ID (SID)
- Two service instances at different places with binding ID BID22 & BID 32

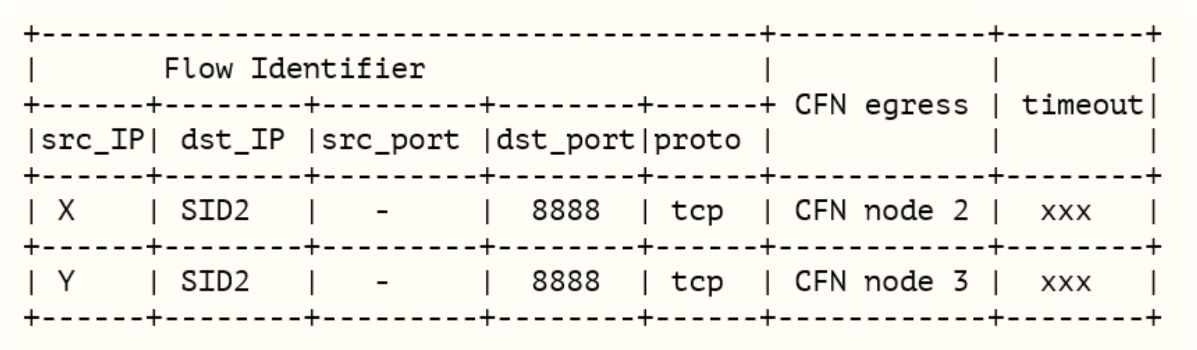
Flow affinity – select the best edge and stick to it



A new flow: selects the most appropriate CFN node and service instance.

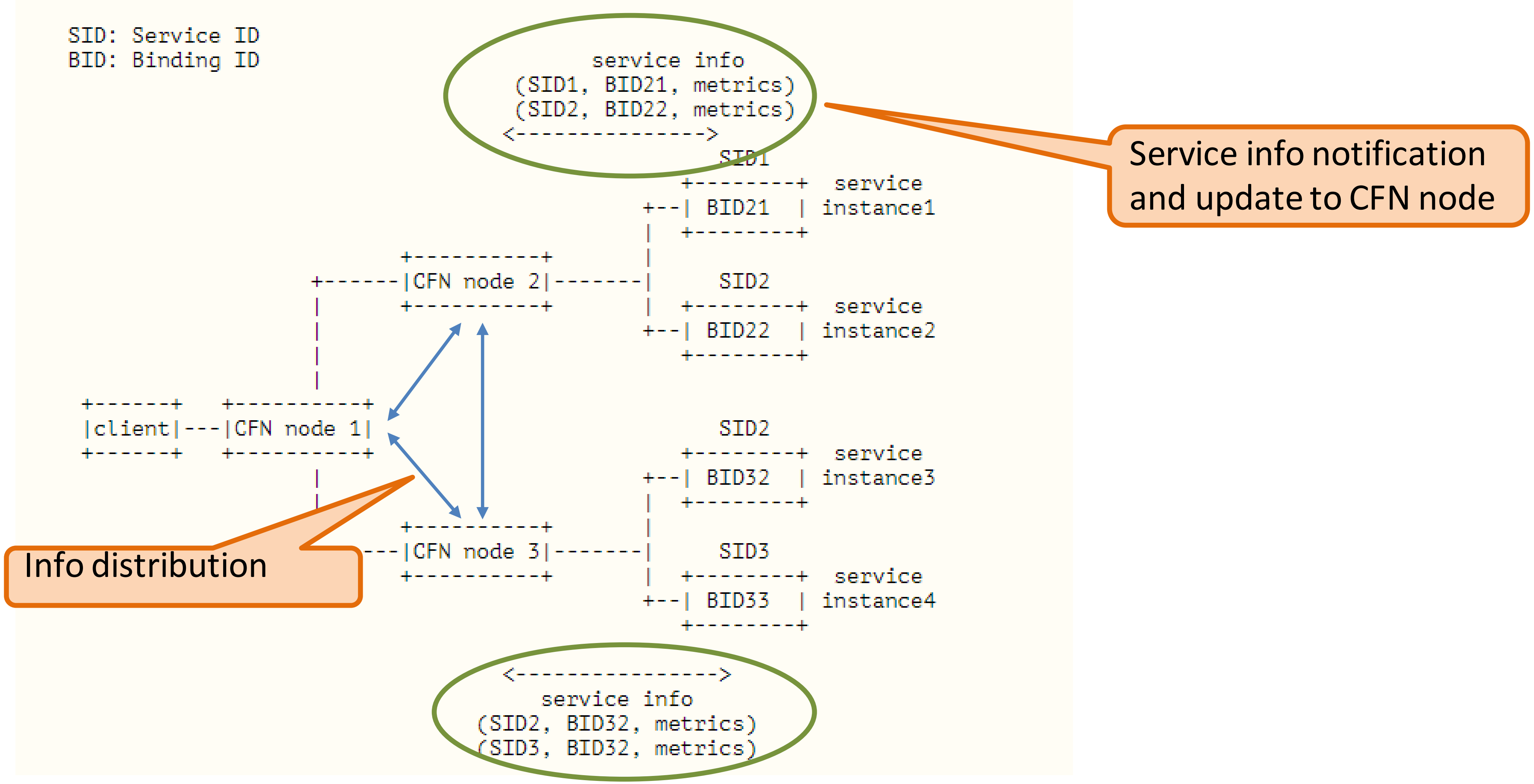
Flow affinity: the subsequent packets of an existing flow are always delivered to the same service instance

Example of flow binding table



Flow Identifier					CFN egress	timeout
src_IP	dst_IP	src_port	dst_port	proto		
X	SID2	-	8888	tcp	CFN node 2	xxx
Y	SID2	-	8888	tcp	CFN node 3	xxx

Computing Aware Routing



```
  SID: Service ID
  BID: Binding ID
                            service info
                        (SID1, BID21, metrics)
                        (SID2, BID22, metrics)
                       <---------------->
                                  SID1
                                +--------+  service
                             +--| BID21  | instance1
                             |  +--------+
              +----------+   |
       +------|CFN node 2|-------|     SID2
       |      +----------+   |  +--------+  service
       |                     +--| BID22  | instance2
       |                        +--------+
       |
 +------+  +----------+
 |client|---|CFN node 1|              SID2
 +------+  +----------+         +--------+  service
       |                     +--| BID32  | instance3
       |                     |  +--------+
       |      +----------+   |
       +---|CFN node 3|-------|     SID3
              +----------+   |  +--------+  service
                             +--| BID33  | instance4
                                +--------+
                       <---------------->
                          service info
                      (SID2, BID32, metrics)
                      (SID3, BID32, metrics)
```

Service info notification and update to CFN node

Info distribution

Summary

- Three features to be supported
  - Anycast based service addressing methodology
  - Flow affinity
  - Computing Aware Routing
- Potential work required
  - Represent computing metrics in defined service/service instance context
  - Distribute the metrics, format and how dynamic/frequent the updates should be
  - Use the metrics in route determination
  - Definition of requirements for any new data plane extensions and procedures.

Your help is welcome

- We will have a virtual side meeting on cfn-dyncast
  - Understand the problem space, gaps and challenges
  - Review the dyncast architecture
  - Discuss the potential work and where to fit them in IETF
- Time: Wed (Nov 18) , 75min, 5 min after plenary ends
  - UTC 10:45 - 12:00
  - CET (UTC+1) 11:45 - 13:00
  - Bangkok Time (UTC+7) 17:45 - 19:00
  - CST (UTC+8) 18:45 - 20:00
  - PST (UTC-8) 02:45 - 04:00
- Webex:
  - Webex Meeting number (access code): 175 335 6387
  - Password:7wrDVwRt7B4
  - Password if joining from a phone: 79738978
  - Webex: https://fipe-meeting.my.webex.com/fipe-meeting.my/j.php?MTID=m2e7d90ec32145ba12f4a6b7e8baf3bcd
- Information also available on side meeting wiki and github:

  https://trac.ietf.org/trac/ietf/meeting/wiki/109sidemeetings

  https://github.com/cfn-dyncast/ietf109