Dynamic Anycast (Dyncast) Side Meeting

IETF 110, Online Meeting
10 March, 2021

Side meeting chairs:
Peng Liu (China Mobile)
Georgios Karagiannis (Huawei)

IETF Note Well

https://www.ietf.org/about/note-well/

This is a reminder of IETF policies in effect on various topics such as patents or code of conduct. It is only meant to point you in the right direction. Exceptions may apply. The IETF's patent policy and the definition of an IETF "contribution" and "participation" are set forth in BCP 79; please read it carefully.

As a reminder:

- By participating in the IETF, you agree to follow IETF processes and policies.
- If you are aware that any IETF contribution is covered by patents or patent applications that are owned or controlled by you or your sponsor, you must disclose that fact, or not participate in the discussion.
- As a participant in or attendee to any IETF activity you acknowledge that written, audio, video, and photographic records of meetings may be made public.
- Personal information that you provide to IETF will be handled in accordance with the IETF Privacy Statement.
- As a participant or attendee, you agree to work respectfully with other participants; please contact the ombudsteam (https://www.ietf.org/contact/ombudsteam/) if you have questions or concerns about this.

Definitive information is in the documents listed below and other IETF BCPs. For advice, please talk to WG chairs or ADs:

BCP 9 (Internet Standards Process)

BCP 25 (Working Group processes)

BCP 25 (Anti-Harassment Procedures)

BCP 54 (Code of Conduct)

BCP 78 (Copyright)

BCP 79 (Patents, Participation)

https://www.ietf.org/privacy-policy/ (Privacy Policy)

NOTE on side meeting

- Open to all
- Meeting minutes will be publicly posted
- Not under NDA of any form
- Please list your full name (*if you are not logged in using your full name*) and affiliation in the chat feature of webex (or CODIMD) during meeting

NOTE on side meeting

- During presentations only clarification questions are welcomed (use +q in the chat)
- Discussions at the end of all presentations (try speak freely)
- Wikipage, including the Webex access information:
  - https://trac.ietf.org/trac/ietf/meeting/wiki/110sidemeetings
- dyncast github: https://github.com/dyncast/ietf110
- CODIMD for notes: https://codimd.ietf.org/notes-ietf-110-dyncast-side-meeting
- *Polling will be done at the end on wrap-up question using the Polling system provided by Webex (Luigi Iannone will assist)*

Agenda

1. Admin [8: 8/120]
2. Problem statement, use cases and requirements (updates): Peng Liu (China Mobile) [15: 23/120]
   - https://tools.ietf.org/html/draft-liu-dyncast-ps-usecases-01
   - https://tools.ietf.org/html/draft-liu-dyncast-reqs-00
3. Dyncast architecture (updates): Luigi Iannone [10 : 33/120]
   - https://tools.ietf.org/html/draft-li-dyncast-architecture-00
4. Questions and Recap of the feedback received[20: 53/120]
5. Specific technical topics – Moderator: Dirk Trossen (37: 90/120)
   - Service affinity (27')
     - Basic approach, aspects to consider – Dirk Trossen (7')
     - Time period based affinity – Carsten Bormann(7')
       - https://www.ietf.org/archive/id/draft-bormann-dyncast-affinity-00.html
     - Discussions, e.g. pros and cons, direction to go, allow multiple solutions for different use case? (13')
   - Metrics discussion (10')
6. Wrap up and next steps (chairs) [20 : 110/120]

Dynamic-Anycast (Dyncast) Scenarios and Requirements

draft-liu-dyncast-ps-usecases-01

draft-liu-dyncast-reqs-00

P. Liu, China Mobile

P. Willis, BT

D. Trossen, Huawei

Major Changes from IETF 109

- Split the ps draft across two drafts:
  - **draft-liu-dyncast-ps-usecases-01** : focus on the use case and problems in existing solutions
  - **draft-liu-dyncast-reqs-00** : focus on the dyncast technical requirements
- Refine the use cases
- Analysis of the deficiency of existing solutions
- Explicitly list the requirements
- Align definitions across documents

- **Definitions**
  - Service
  - Service instance
  - Service identifier
  - Anycast
  - Dyncast: Dynamic Anycast

Recap

What does user want?

- **The best user experience such as low latency and high reliability, etc .**

How to deploy in order to meet the requirements?

- **Providing Functional Equivalency**
  - Deploy instances for the same service across edge sites for better availability
- **Steer Traffic Dynamically to the "Best" Service Instance**
  - Traffic is delivered to optimal edge sites according to networking/computing status (defining what 'best' is for each service)

Challenges

- Geographically Scattered Large Number of Edge Sites
  - Highly distributed
  - May or may not have proximate distances to user
- Resource Limitation
  - *fewer servers – 10s of server per node.*
- Heterogeneous Hardware
  - *CPU, GPU, Memory, ASICs*
- Dynamic Load
  - *Available resources change quickly*
- Edge-cloud Coordination
  - *Edge does not solve all*
- High Cost
  - *On-site maintenance is expensive*
- Mission Critical
  - *Users are counting on you (i.e. 100% reliability of industry automation)!*

Many of these challenges are NOT solvable solely in "Computing Domain" NOR the "Network Domain" alone.

Can we have a collaborative approach?

Updates

Use Case – AR/VR

Upper bound latency for motion-to-photon(MTP):to less than **15-20 ms** to **avoid the motion sickness**

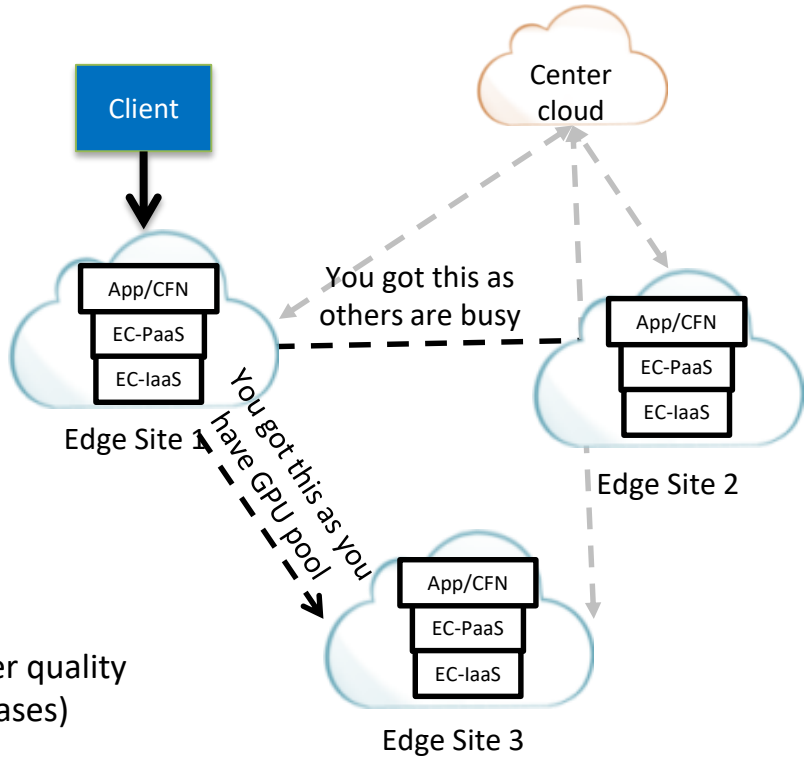
a) sensor sampling delay: <1.5ms
b) display refresh delay: ≈5 ms
c) frame rendering computing delay with GPU≈ **5.5ms**
d) network delay =20-1.5-5.5-7.9 = **5.1ms**

The budgets for computing delay and network delay are almost equivalent!!

Computing resources have a big difference in different edges

Examples for need to dynamically steer traffic to the 'best' edge:

- Training in center cloud, whilst detection in edge DC
- Rendering tasks need to be diverted to GPU infrastructure for better quality
- Traffic/compute offloading for tide effect (Theatre/Sport stadium cases)



Deficiency in existing solutions

- **Dynamicity of Relations:** Existing solutions exhibit limitations in providing dynamic 'instance affinity'
  - E.g., DNS is not designed for this level of dynamicity (i.e., minute level originally, client needs to flushing the local DNS cache, frequent resolving may lead to overload of DNS)
- **Efficiency:** Existing solutions may introduce additional latencies and inefficiencies (e.g., additional path stretch & more messages) in packet transmission
- **Complexity:** Existing solutions require careful planning for the placement of necessary control plane functions in relation to the resulting data plane traffic
- **Metric exposure and use:** Existing solutions lack the necessary information to make the right decision on the selection of the suitable service instance due to the limited semantic or due to information not being exposed

- **Security**: Existing solutions may expose control as well as data plane to the possibility of a distributed Denial-of-Service attack on the resolution system as well as service instance.
- **Infra changes**: Existing solutions require changes to service and/or network infrastructure, with no solution limiting the necessary changes to the very ingress point of the network

Requirements for Dyncast Architecture

- **Anycast-based Service Addressing Methodology**
  - Mapping of a unique service identifier to specific unicast address
- **Instance Affinity**
  - Maintain "instance affinity" which MAY span one or more service requests, i.e., all the packets from the same flow MUST go to the same service instance.
- **Encoding Metrics**
  - Agree on the service-specific metrics, while obfuscating the specific semantic to preserve privacy
- **Signaling Metrics**
  - Signal the metrics for using in routing decisions, realize means for rate control and avoid loop
- **Using Metrics in Routing Decisions**
  - Specify default action (associated to service identifier) to be taken with possible alternative action
- **Supporting Service Dynamism**
  - Support possibly highly dynamic affinity relations, possibly down to single requests

Summary

- Service providers are offering the performance of the integrated computing and networking infrastructure as a quality of experience to their customers
- **Problem**: How to optimally route service demands based on service and network metrics to the best edge?
- Existing IETF protocol specification work does not sufficiently solve the identified problem at the network level
  - **Exposing up-to-date knowledge of computing resources to the network layer -> Service and network metrics collection, representation, distribution and how to use them for edge determination**

Agenda

1. Admin [8: 8/120]
2. Problem statement, use cases and requirements (updates): Peng Liu (China Mobile) [15: 23/120]
   - https://tools.ietf.org/html/draft-liu-dyncast-ps-usecases-01
   - https://tools.ietf.org/html/draft-liu-dyncast-reqs-00
3. Dyncast architecture (updates): Luigi Iannone [10 : 33/120]
   - https://tools.ietf.org/html/draft-li-dyncast-architecture-00
4. Questions and Recap of the feedback received[20: 53/120]
5. Specific technical topics – Moderator: Dirk Trossen (37: 90/120)
   - Service affinity (27')
     - Basic approach, aspects to consider – Dirk Trossen (7')
     - Time period based affinity – Carsten Bormann(7')
       - https://www.ietf.org/archive/id/draft-bormann-dyncast-affinity-00.html
     - Discussions, e.g. pros and cons, direction to go, allow multiple solutions for different use case? (13')
   - Metrics discussion (10')
6. Wrap up and next steps (chairs) [20 : 110/120]

Dyncast Architecture
(dynamic anycast)

draft-li-dyncast-architecture

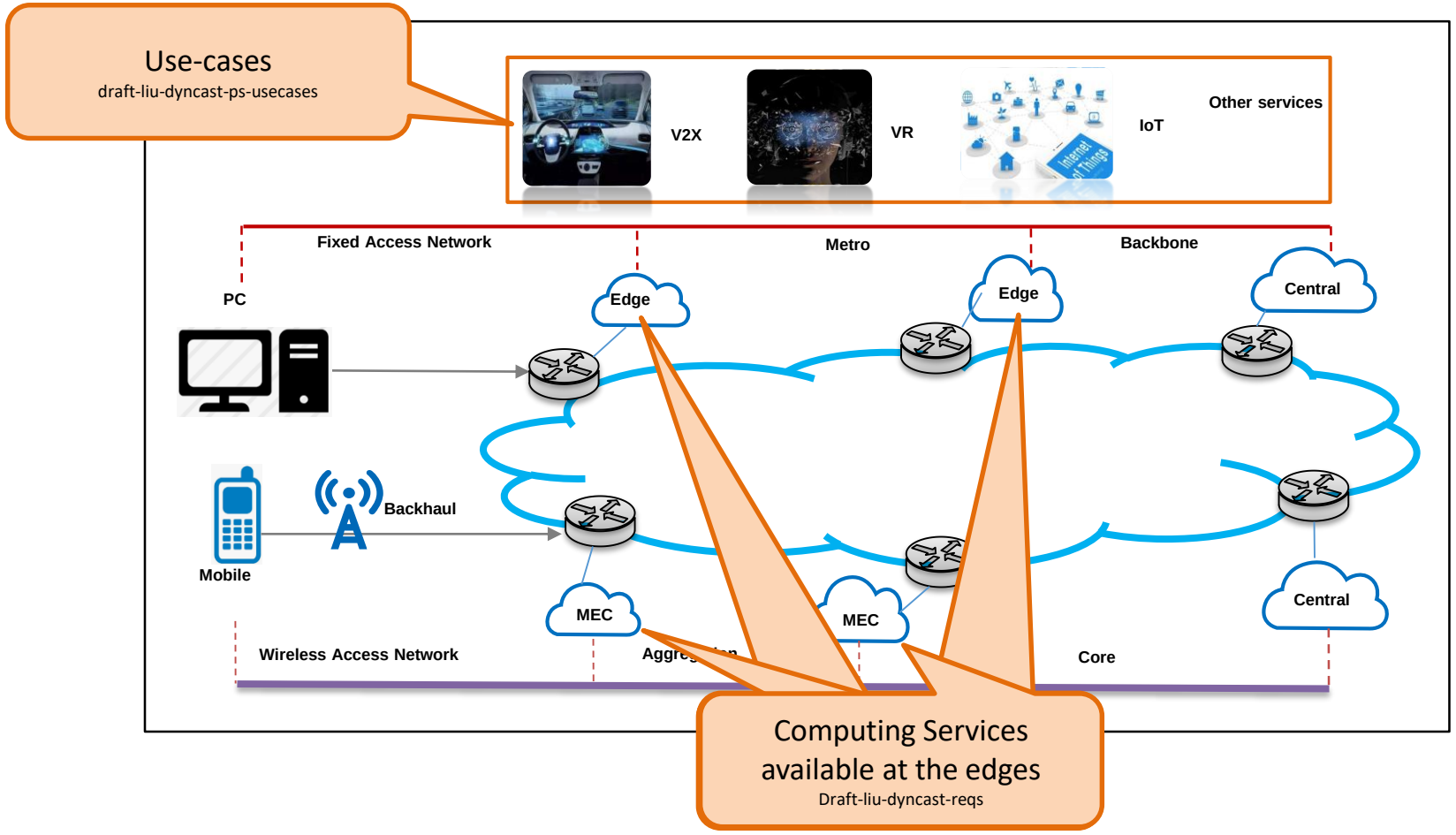
Yizhou Li; Luigi Iannone; Dirk Trossen; Peng Liu

110 IETF Side Meeting

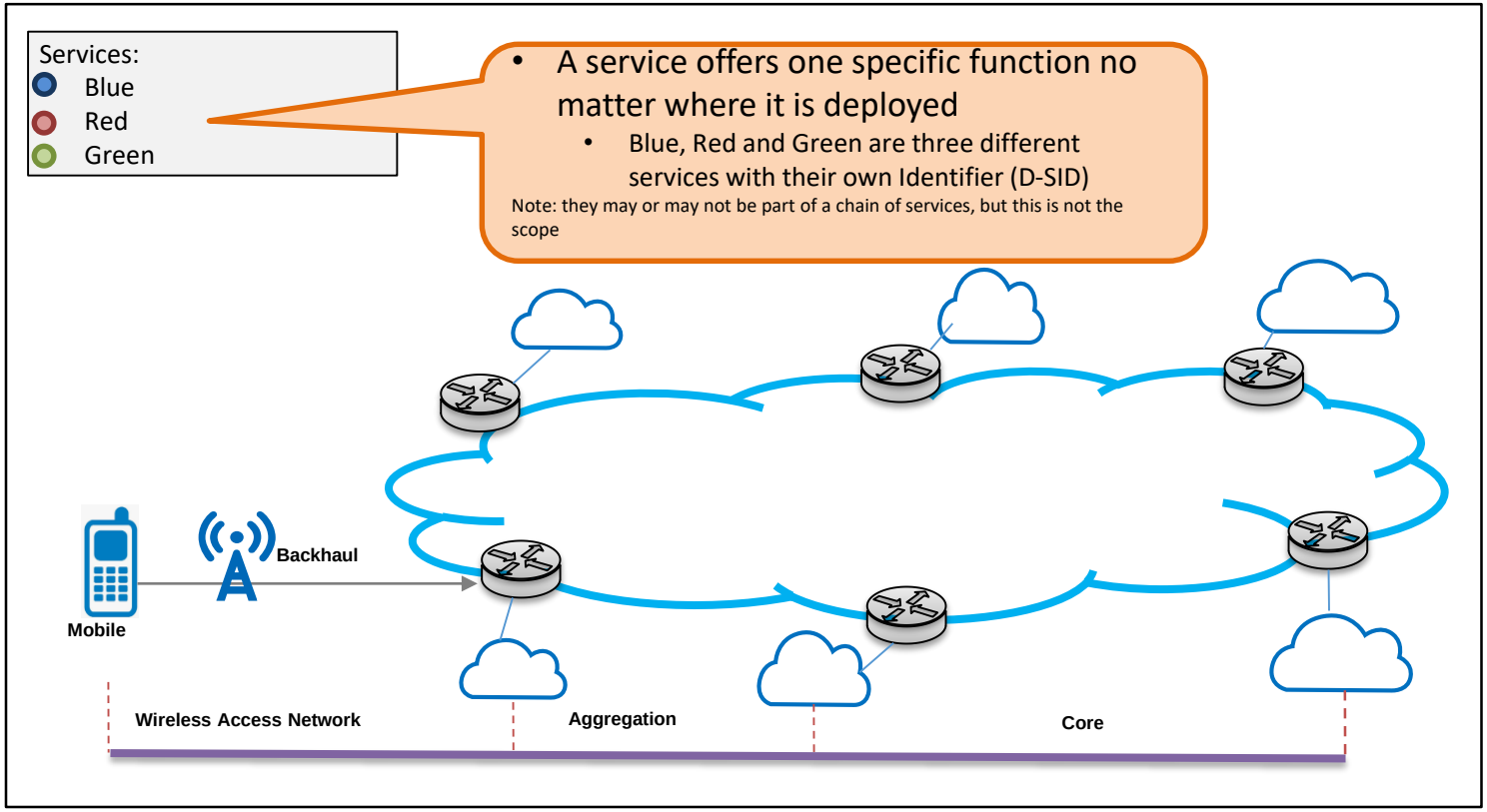
Major Changes from IETF 110

- TL;DR
  - Everything
- Revised Terminology
  - Architectural elements:
    - D-Router: Dyncast Router
    - D-MA: Dyncast Metric Agent
    - D-Forwarder: Dyncast Forwarder
  - Architectural Parameters:
    - D-SID: Dyncast Service ID
    - D-BID: Dyncast Binding ID
- Revised (almost all) text
  - Adapted text to new terminology
  - Improved description of Dyncast Workflow
  - Clarified Control- vs Data- plane operations

Context

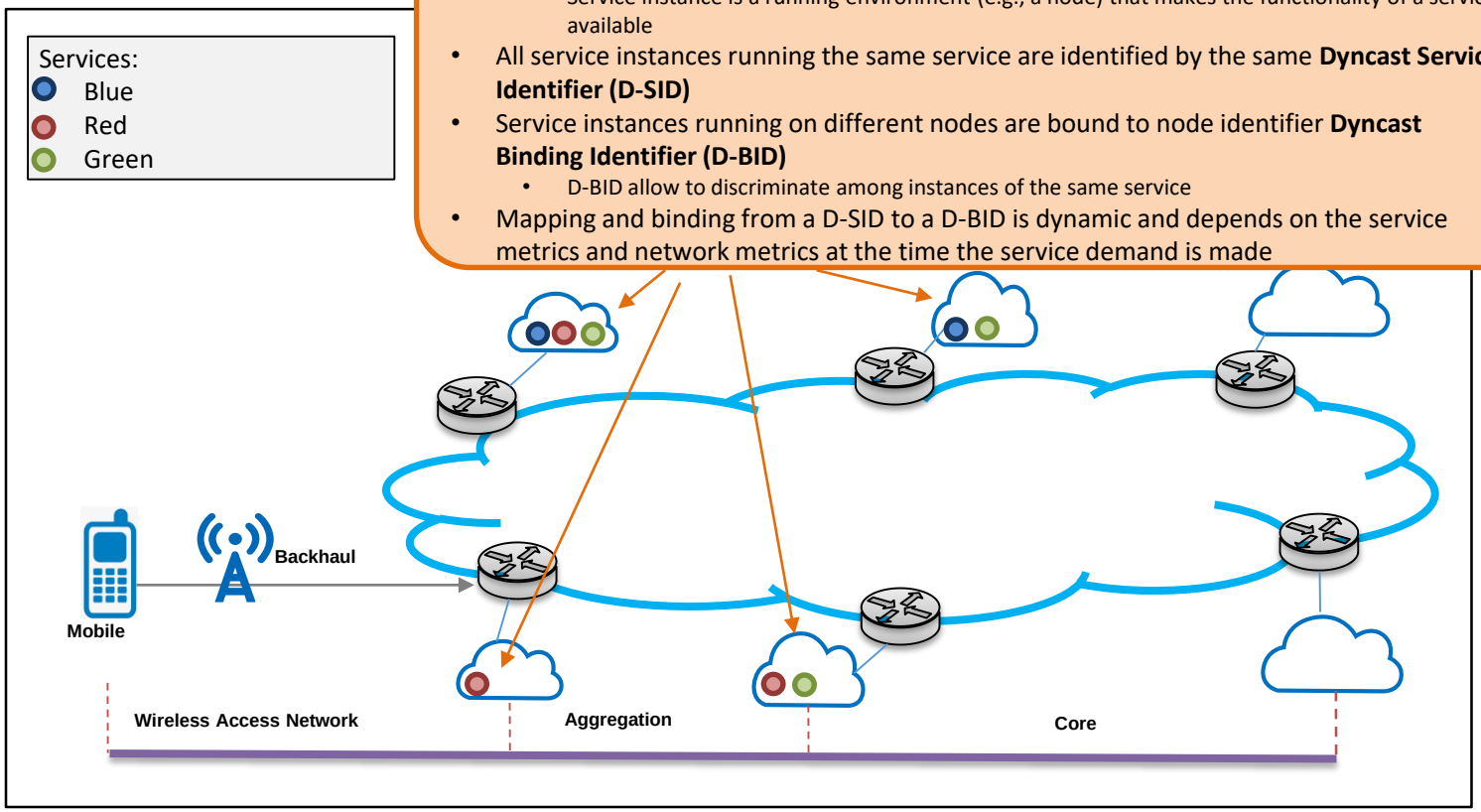

Use-cases
draft-liu-dyncast-ps-usecases
V2X
VR
IoT
Other services
Fixed Access Network
Metro
Backbone
PC
Edge
Edge
Central
Backhaul
Mobile
MEC
MEC
Central
Wireless Access Network
Core
Computing Services
available at the edges
Draft-liu-dyncast-reqs

Services & Service Instances

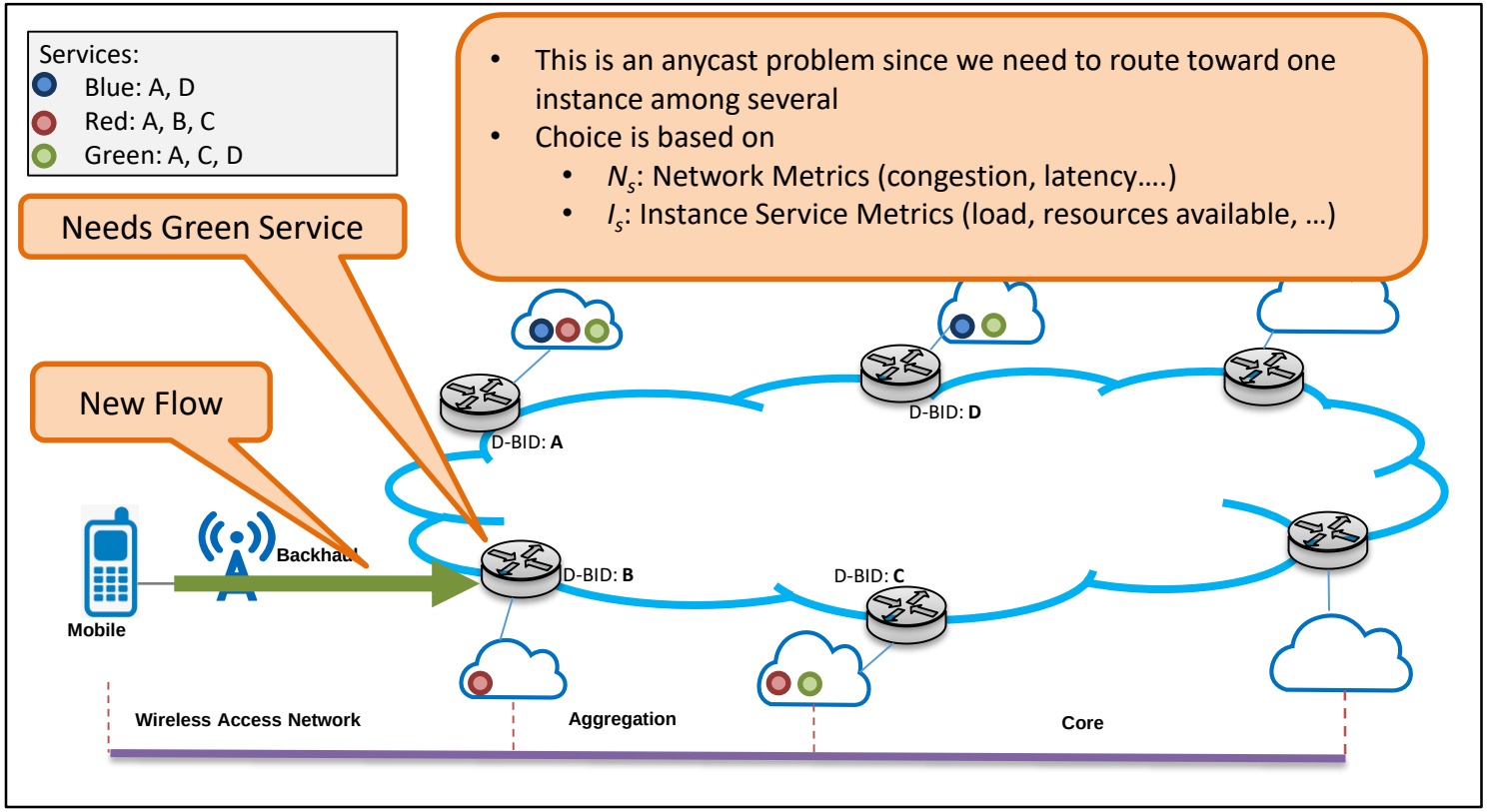


Services & Service Instances

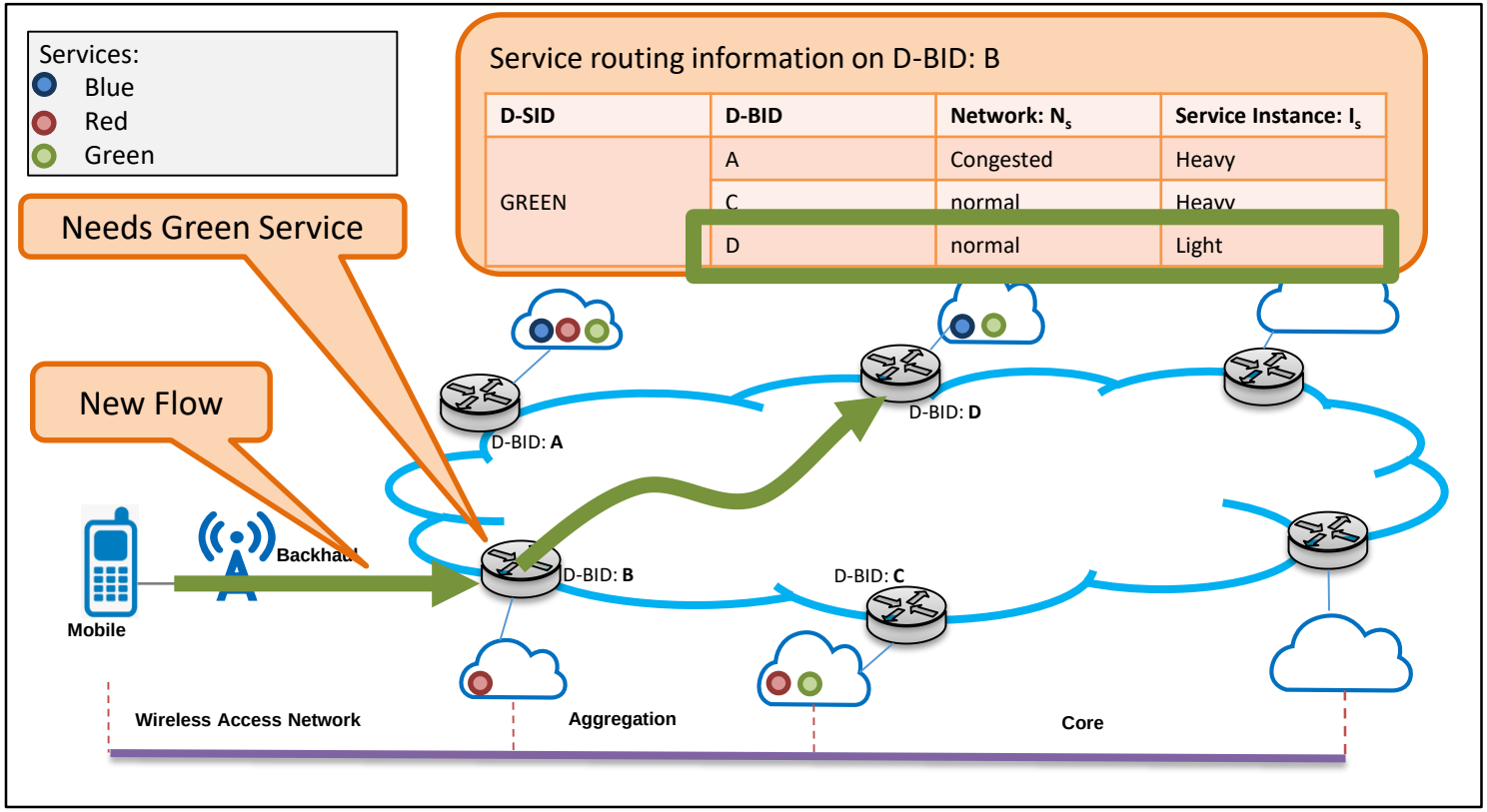
- One Service can have several instances running on different nodes
  - Service instance is a running environment (e.g., a node) that makes the functionality of a service available
- All service instances running the same service are identified by the same **Dyncast Service Identifier (D-SID)**
- Service instances running on different nodes are bound to node identifier **Dyncast Binding Identifier (D-BID)**
  - D-BID allow to discriminate among instances of the same service
- Mapping and binding from a D-SID to a D-BID is dynamic and depends on the service metrics and network metrics at the time the service demand is made



Service Demand Dispatch



Service Demand Dispatch

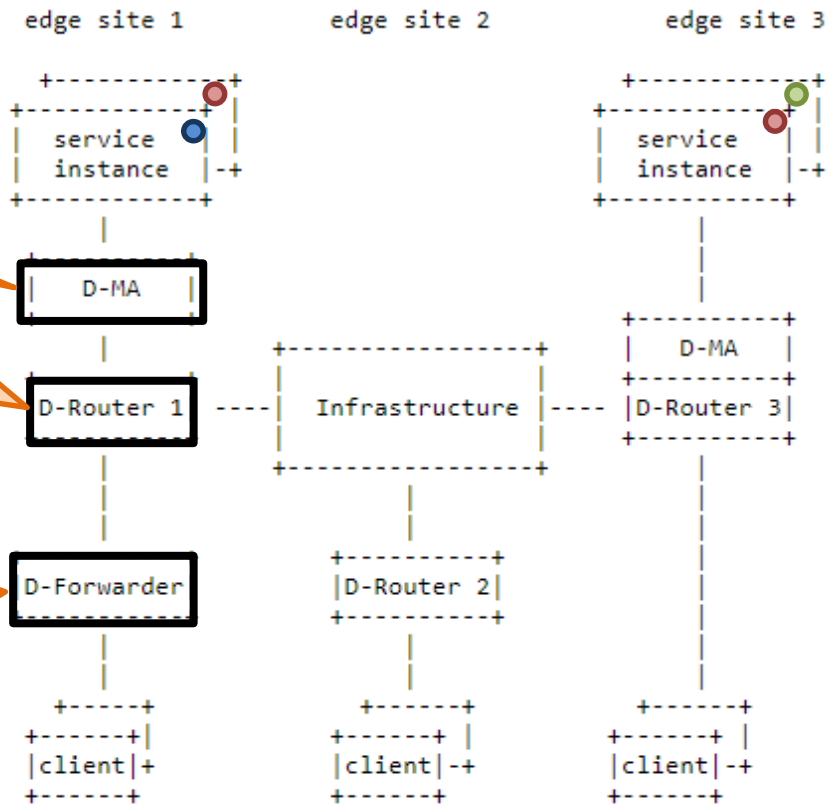


Dyncast: Architecture

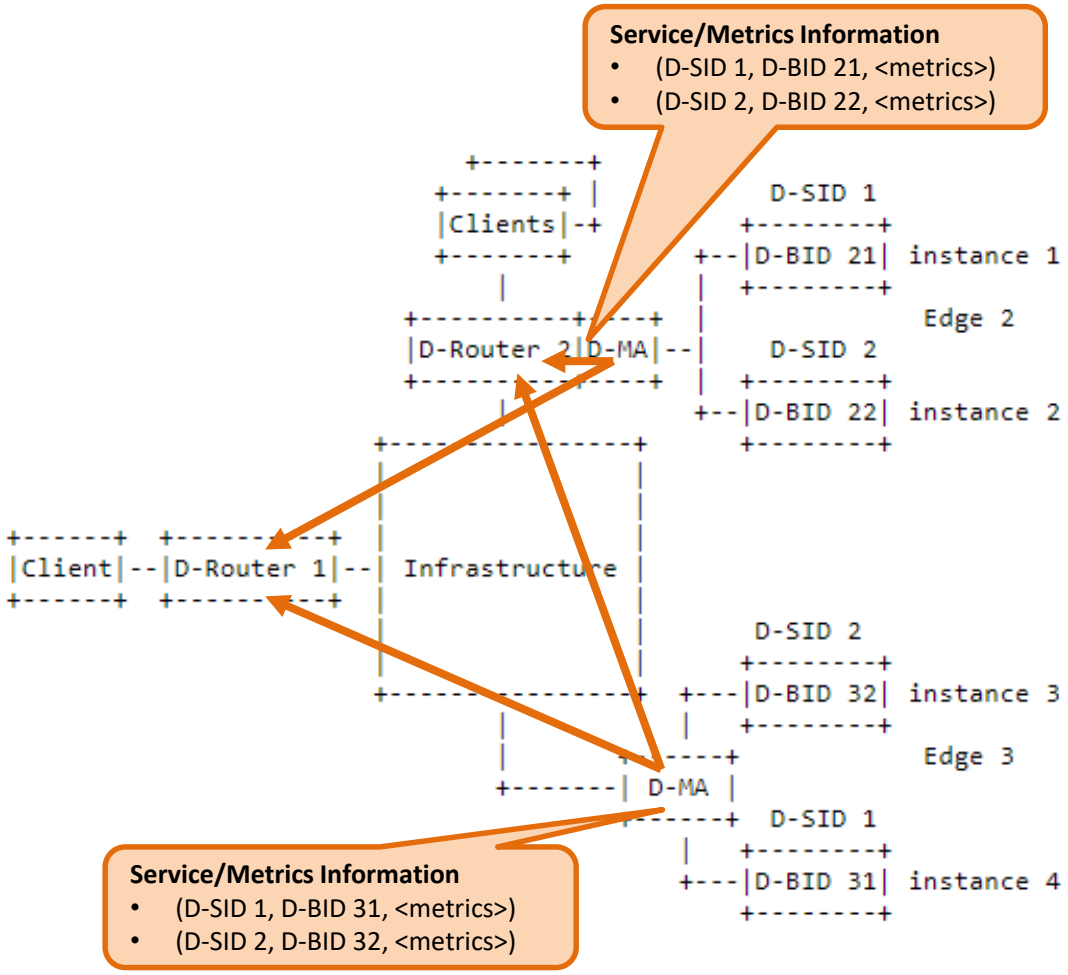
- **Dyncast Metric Agent (D-MA):** A Dyncast specific agent able to gather and send metric updates (from both network and instance respective) but not performing forwarding decisions. May run on a D-Router, but it can be also implemented as a separate module (e.g., a software library) collocated with a service instance.

- **D-Router:** A node supporting Dyncast functionalities. It is able to understand both network-related and service-related metrics, take forwarding decision based upon and maintain instance affinity, i.e., forwards packets belonging to the same service demand to the same instance when belonging to the same flow.

- **D-Forwarder**: An optional element able to forward packets towards a service instance, while not receiving any metric and as such not being able to make any decision when a new service demand arrives. It relies on a D-Router for the decision, it only guarantees instance affinity.

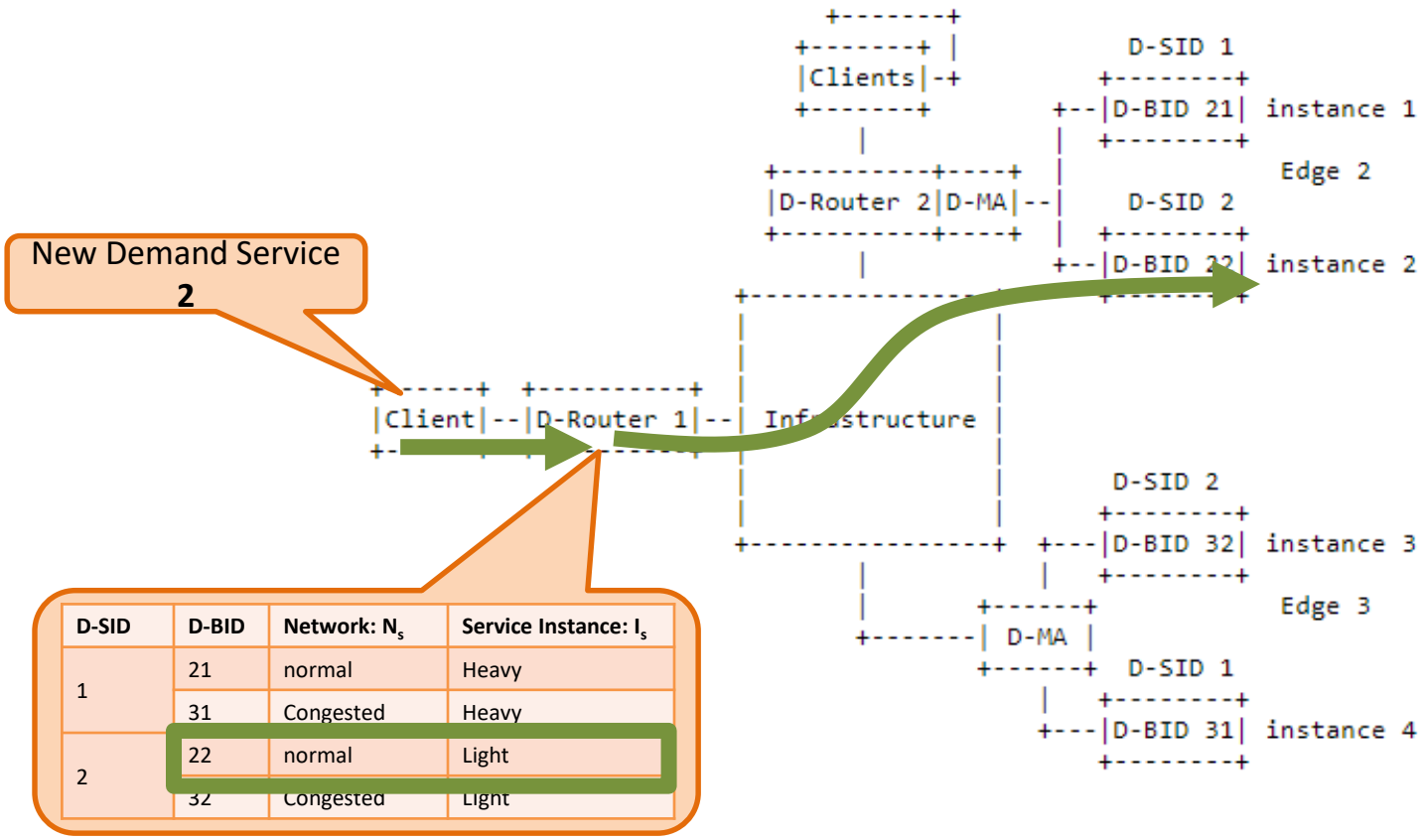


D-MA: Dyncast Metric Agent



- Note: Arrows indicate how information flows not actual communication sessions

D-Router



D-SID	D-BID	Network: N_s	Service Instance: I_s
1	21	normal	Heavy
	31	Congested	Heavy
2	22	normal	Light
	32	Congested	Light

Service Affinity & Binding Table

- Service affinity is one of the critical features that Dyncast should support
- Binding table allows to determine the most appropriate egress and service
- An entry in the binding table can be identified using the classic 5-tuple value
  - different services may have different granularity of flow identification

More Next

Flow Identifier					BID	timeout
src_IP	dst_IP	src_port	dst_port	proto		
X	SID2	-	8888	tcp	BID22	xxx
Y	SID2	-	8888	tcp	BID32	xxx

Summary

- Consolidated Architecture
  - Coherent terminology
  - Well defined architectural elements
- Questions:
  - Is there anything missing?
  - Is there anything to be dropped?
- Further refinements needed
  - Help very welcome ☺
  - Initial feedback positive but text need further refinement

Agenda

1. Admin [8: 8/120]
2. Problem statement, use cases and requirements (updates): Peng Liu (China Mobile) [15: 23/120]
   - https://tools.ietf.org/html/draft-liu-dyncast-ps-usecases-01
   - https://tools.ietf.org/html/draft-liu-dyncast-reqs-00
3. Dyncast architecture (updates): Luigi Iannone [10 : 33/120]
   - https://tools.ietf.org/html/draft-li-dyncast-architecture-00
4. Questions and Recap of the feedback received[20: 53/120]
5. Specific technical topics – Moderator: Dirk Trossen (37: 90/120)
   - Service affinity (27')
     - Basic approach, aspects to consider – Dirk Trossen (7')
     - Time period based affinity – Carsten Bormann(7')
       - https://www.ietf.org/archive/id/draft-bormann-dyncast-affinity-00.html
     - Discussions, e.g. pros and cons, direction to go, allow multiple solutions for different use case? (13')
   - Metrics discussion (10')
6. Wrap up and next steps (chairs) [20 : 110/120]

Feedback received after last meeting

- Please check github for the full question list document. https://github.com/dyncast/ietf110
- Some questions received are instance affinity or metrics related. We will discuss them in next agenda item. Please hold such questions for now.

Questions and Discussion

Agenda

1. Admin [8: 8/120]
2. Problem statement, use cases and requirements (updates): Peng Liu (China Mobile) [15: 23/120]
   - https://tools.ietf.org/html/draft-liu-dyncast-ps-usecases-01
   - https://tools.ietf.org/html/draft-liu-dyncast-reqs-00
3. Dyncast architecture (updates): Luigi Iannone [10 : 33/120]
   - https://tools.ietf.org/html/draft-li-dyncast-architecture-00
4. Questions and Recap of the feedback received[20: 53/120]
5. Specific technical topics – Moderator: Dirk Trossen (37: 90/120)
   - Service affinity (27')
     - Basic approach, aspects to consider – Dirk Trossen (7')
     - Time period based affinity – Carsten Bormann(7')
       - https://www.ietf.org/archive/id/draft-bormann-dyncast-affinity-00.html
     - Discussions, e.g. pros and cons, direction to go, allow multiple solutions for different use case? (13')
   - Metrics discussion (10')
6. Wrap up and next steps (chairs) [20 : 110/120]

Discussion

Instance Affinity – WHAT and HOW?

Instance Affinity – WHAT?

- **Instance affinity** is the binding of a specific client to a specific service instance for given D-SID, the instance represented by D-BID

- Dyncast solution MUST maintain "instance affinity" which MAY span one or more service requests, i.e., send all related packets to same server instance

- **Service invocation** is a set of packets that would benefit from instance affinity

Objectives

- Make sure packets from one service invocation go to same instance
  (Affinity is not really a guarantee, but best effort)
- Avoid adding roundtrips for instance selection:
  Don’t run a separate resolution process in the application, **just send**
- Enable progress:
  Enable packets from a new service invocation to go to better instance

Instance Affinity - HOW? Option 1

Explicit client state in d-forwarder (or d-router):

- Use explicit affinity marking , i.e., the signaling of the start/end of a service invocation, store D-SID->D-BID binding for client at d-forwarder (or d-router), and use the ephemeral state later if affinity marking persists.
- **Pro:**
  - client/app decides on affinity
  - No specific affinity state in d-router
- **Con**:
  - Client's affinity state in d-forwarder (or d-router, if no d-forwarder).
  - Need to find that marking
    - Could be a bit (could use something similar to option 5 in a manner)
    - Could be 'flow identifier' in packet (e.g., 5 tuple)
  - Needs client change and possibly per-app mechanism

Instance Affinity - HOW? Option 2

Leave it to app level protocol:

- Service demand is ALWAYS mapped according to latest metric values.
- Service instance could signal D-BID back to client, which is then creating a direct service request to the D-BID for the duration of the transaction

  -> the affinity handling is entirely pushed to the application.

  -> the first packet would decide on affinity (e.g., within the initial QUIC or TCP packet) by sending back D-BID or not
- **Pro:** App decides on affinity
- **Con**:
  - Exposes D-BID to client, which may not be wanted by service, e.g., for security reasons
  - Needs client change and possibly per-app mechanism

Instance Affinity - HOW? Option 3

Signaling of D-BID mapping to client at net level:

- Service demand is mapped in d-forwarder/d-router, turned into a service request to service instance
- Chosen D-BID is signaled back to client with explicit CP message.
  - D-BID is used by dyncast 'stack' at client when affinity bit is being set in API, otherwise a service demand is being created instead.
- **Pro:** does not rely on affinity markings, as in option 1
- **Drawback**:
  - additional signaling from d-forwarder/d-router to client (possibility of loss, which may break the transaction as a consequence)
  - Exposes D-BID to client, which may not be wanted by service, e.g., for security reasons
  - Needs client change

Instance Affinity - HOW? Option 4

Integrate affinity into metric-based decision:

- D-Router makes mapping decision from D-SID to suitable D-BID
- Affinity can be seen treated as another (service-specific) metric input into that decision
- **Pro**:
  - no explicit/separate signaling of affinity
  - No signaling of D-BID mappings
  - Part of explicit routing decision makes it part of DP operations
  - Needs no client change
- **Con**:
  - May need explicit affinity state
  - Affinity at app level may not be visible at routing level
  - Client state on the D-Router if affinity is demanded
  - Client state also extends to D-Forwarder (unlike a D-SID->S-BID mapping without affinity)

Instance Affinity - HOW?
Option 5

dyncast instance affinity:
Averting per-application state via periodic state updating

Carsten Bormann, dyncast IETF110 side meeting, 2020-03-10

Assumptions

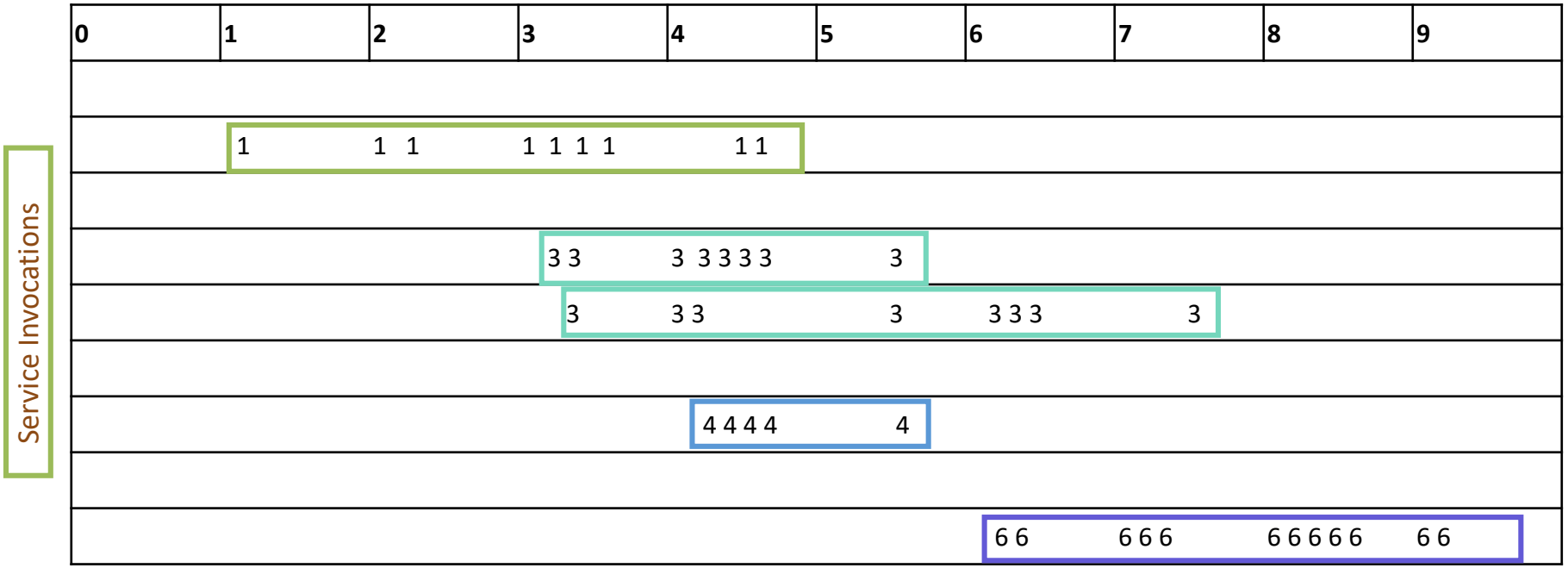
- There needs to be a way for the client to make different service invocations distinguishable to the network
  - Potential assumption A: Can be done via 5-tuple
    Requires the application to make new “connections” per invocations
  - Our assumption B: Application is willing to explicitly indicate separation
- Objective: Avoid application state in the network ---
  no state in network about every single client and their service invocations
- Duration of service invocation (service *stretch*) is a few minutes max
  - For longer invocations, separate lookup roundtrips would be in the noise
- Network path optimization updates not more often than every few seconds
- Some loose (~ 2–3 s) time synchronization is achievable (applications, network)

Approach

- Rhythm: Divide time into periods p (say, 10 seconds each)
  - Due to time synchronization, applications and network are aware of approximate current period
  - Updates in instance selection optimization are becoming active each period
- In each packet for a service invocation,
  client signals the period number that was current when invocation started
  - Using destination address for this signaling
  - Limited number of bits (say, b=6), cyclic reuse after 2^b periods (say ~10 min)
  - Service stretch $s < 2^b$ periods
- Network keeps $N = s/p + \varepsilon$ versions of the optimization state (say, 33)

Service invocations and period numbers

Each packet bearing invocation's initial period number



Discussion (+ ° –)

+ Averted: ~~Keep (and possibly disseminate) per-invocation state in the network~~ ✓
+ Naïve routers do the right thing; naïve clients just get slightly outdated optimization
° Cost: Keep ~ 33 (N = s/p + ε) copies of per-service optimization state in network
  + Might compress well, if most optimization state doesn't change much, or round-robins
– Doesn't address legacy applications:
  ° Period number could be provided in usual DNS address lookup (fallback only)
– Invocations lasting longer than service stretch could lose affinity:
  ° Can be addressed via a time-based retest/migration approach
  ° Or will simply occasionally be cut short
– IPv4: Where to put the 6 bits?
  ° Could be done in destination port (➔ 48-bit address)

➔Slightly sharpening the affinity definition leads to a solution with different features and bugs.

Is this a better direction? **Are there other better directions?**

Discussion

Metrics – Scoping Possible Work

Metrics: Early Discussion

- What examples of metrics do people have in mind?
  - How encoded?
  - Frequency of updates?

- What algorithms do people have in mind for using those metrics?
  - How encoded and realized?

Agenda

1. Admin [8: 8/120]
2. Problem statement, use cases and requirements (updates): Peng Liu (China Mobile) [15: 23/120]
   - https://tools.ietf.org/html/draft-liu-dyncast-ps-usecases-01
   - https://tools.ietf.org/html/draft-liu-dyncast-reqs-00
3. Dyncast architecture (updates): Luigi Iannone [10 : 33/120]
   - https://tools.ietf.org/html/draft-li-dyncast-architecture-00
4. Questions and Recap of the feedback received[20: 53/120]
5. Specific technical topics – Moderator: Dirk Trossen (37: 90/120)
   - Service affinity (27')
     - Basic approach, aspects to consider – Dirk Trossen (7')
     - Time period based affinity – Carsten Bormann(7')
       - https://www.ietf.org/archive/id/draft-bormann-dyncast-affinity-00.html
     - Discussions, e.g. pros and cons, direction to go, allow multiple solutions for different use case? (13')
   - Metrics discussion (10')
6. Wrap up and next steps (chairs) [20 : 110/120]

Any other questions?

Results of Wrap-up discussions during IETF109 side meeting

- Is the problem space clearly defined?
  - A. Yes : 26/48 (54%)
  - B. No: 5/48 (10%)
- Should this work be done in IETF?
  - A. Yes: 28/47 (60%)
  - B. No: 1/47 (2%)
- Is there interest to participate in this work?
  - A. Yes: 25/46 (54%)
  - B. No: 1/46 (2%)

Question to start wrap up

Focus of work: What IETF protocol specification work needs to be done for dyncast?

Dyncast could focus on:

1) Specification of dyncast framework and functional components

2) Reuse existing IETF protocols when possible. Define protocol extensions when needed or introduce a new protocol when necessary including features like:
   - Represent service specific metrics like computing metrics in defined service/service instance context
   - Distribute the metrics using routing protocol, potential impact to the protocol like when metrics updates should be sent
   - Use metrics in route determination
   - Definition of requirements for any new data plane extensions and procedures to ensure the instance affinity;

Wrap-up Discussions and Next Steps

- Are the use cases relevant and compelling?
- Are the identified problems/issues worth to be endorsed and worked out in the IETF Routing Area?
- Is there interest to work towards a charter and BOF proposal?

Polling will be done for each wrap-up question using the Polling system provided by Webex (Luigi Iannone will assist)