Edge Transport (ETRA): Edge Transport Protocol Architecture for Next Generation Mobile IoT Systems

Ravishankar Ravindran†, Aytac Azgin†, and K. K. Ramakrishnan ‡
† Futurewei Technologies, Santa Clara, CA, USA. ‡ CSE, UC Riverside, CA, USA.
†{ravi.ravindran, aytac.azgin}@futurewei.com, ‡kk@cs.ucr.edu

***Abstract*—Next generation mobile IoT systems such as autonomous vehicles, mobile robotic and drone systems demand new ways to integrate programmable edge compute and networking resources to manage and control them in real time. Current solutions to support real-time information delivery and sharing, handling mobility, multicasting and service migration need to evolve. In this paper, we look at the unique features of these IoT systems that take advantage of high bandwidth wireless communications and edge computing. To address these challenges, we examine the capabilities needed in a new transport architecture for edge networks (ETRA) to handle low latency (and high bandwidth) real-time data streams from mobile IoT systems that also require services to handle mobility, service migration, data replication and reliability. We exemplify ETRA's functionality considering the case for augmented vehicular reality, wherein autonomous vehicles share their sensors' point clouds with each other over a managed edge infrastructure.**

I. Introduction

Edge computing has gained significant attention because of the promise of responsive services that take advantage of proximity and increased bandwidth, and more importantly allowing contextual processing of data. While there are still significant issues to be addressed related to ubiquitous deployment of edge compute platforms that support mobility, the increasing demand from the new class of networked IoT systems, such as autonomous vehicles, drone systems and mobile robotic systems is expected to drive these deployments. However, several challenges remain, especially in terms on providing low latency, secure sharing of information, mobility and service migration. Our paper takes a first look at some of these challenges and proposes a new transport architecture that attempts to provide several fundamental building blocks to meet the needs of such networked IoT systems.

The mobility, latency and scale (both in bandwidth, number of nodes and compute capability) needed for such systems may be more stringent compared to AR/VR and other applications that are currently being designed on top of the TCP/IP protocol suite and requires us to re-examine if it is appropriate to consider a new transport architectures. We thus begin by highlighting the unique features of these future IoT systems for which a new transport design is being proposed:

Autonomous Vehicles (AV): Significant effort is being spent in building self-driving vehicular systems, with the assumption that the required compute resources are local to the vehicle. These compute resources are used to process the high bandwidth data generated by on-board cameras, light detection and ranging devices (LIDARs), radars and/or other 3D sensors along with any other control sensory input to help the on-board inference engine with localization, object detection and tracking. The amount of resources and requirements associated with such data and its processing creates a significant limitation, leading to unexpected outcomes. Additionally, this approach of solely depending on on-board computing capability does not fully exploit the potential benefits of having a connected vehicular environment where the data from multiple vehicles can be acted upon for time-critical vehicular and human safety [1].

Drone Systems (DS): DS [2], [3] have many applications ranging from taking part in rescue missions in disaster situations, to delivering media content in customized formats as in using onboard drone cameras covering sports, and to being part of large fleets used by enterprises to deliver packages at large scale. Drone systems require global planning that works over both larger and smaller timescales to provide real-time control from the network edge to help with device level inference and to control directional changes, velocity etc. Further challenges are introduced when a group of drones have to be coordinated to achieve specific tasks such as tracking a specific mobile entity on the road, where optimization objectives could include minimizing the data uploaded using an edge intelligence that offer real-time predictive feedback based on the trajectory history of the target [4].

Mobile Robots (MR): The potential of opening up the robot interfaces to the network offers the opportunity to use the state of the art learning algorithms for customizable behaviours and flexible task assignments, in different domains. Particularly situations where large scale MR systems, such as robotic systems developed by Boston Dynamics [5], work in a coordinated manner to substitute external human tasks would benefit from edge support to achieve high efficiency and safety. In a connected environment, a robot's data should be securely transferred to the edge compute platform, from where more refined behavioral models may then be sent to the devices [6].

To support these systems in large scale, we offer a new data-centric transport protocol design, referred to as the Edge Transport (ETRA), which carries the following components to help with large scale resource-hungry and time-sensitive IoT deployment: (i) functionality to enable efficient named data streams (NS)[1] sharing among distributed producers, consumers and edge services (ii) per-session caches in the transport layer, which is helpful during mobility and while operating over an end-to-end secure session, (iii) a service-scoped transport layer resolution function, which resolves NS to end point identifiers, and (iv) a cache migration system, which is used when edge service migration is required.

In short, in this paper, we lay down the architectural details of our new transport protocol by elaborating on its different

[1]Named data streams refer to the stream of named content generated by a producer under a name prefix. For e.g. LIDAR data stream from an AV could be named as / <AV-ID> / <LIDAR> /∗

functional components and its end-to-end protocol operation. We start by presenting the system model and the requirements for these next generation IoT systems in Section II. We discuss the requirements and design principles for the new edge transport in Section III. We present the details of our protocol and its operation in Section IV, and conclude our discussions in Section V.

II. System Model and Networking Requirements

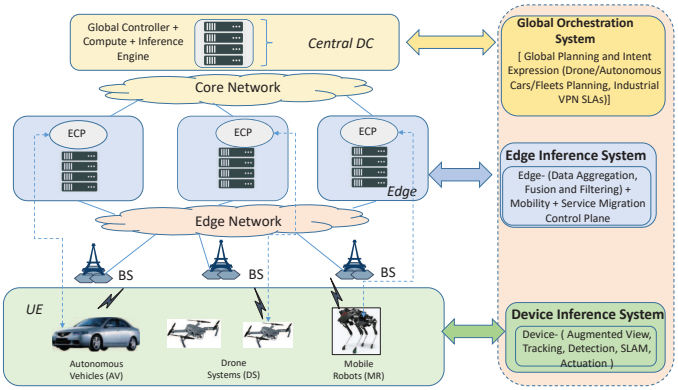

Fig. 1. System model for edge compute and network platform.

We show the system level architecture in Figure 1, which depicts a typical view of systems requiring edge compute and storage support. In this three tier architecture, at the lowest level, we have the user entity (UE), which has compute resources to process its sensor data and to execute several inference tasks related to object detection, tracking, localization and mapping. At the second level, we have the compute edge platform, which consists of edge control points (ECPs) and which is placed no further than the central office (CO). Compute edge platform offers the benefits of both offloading the local compute as well as generating complimentary network view (which can be fused with crowdsourced input from other UEs in its vicinity, and other static resources around its location) that can be input to the UE's inference engine in real time to improve the decision making of the UE's inference engine. In this architecture, we assume a one-to-one relationship between the ECP and the UE, considering the need for performance and predictability towards the workloads. At the third level, we have the central cloud, which is involved in longer time scale compute and network provisioning to sustain the edge services.

The above systems and associated frameworks differ from the traditional, and mostly, stationary and low bandwidth IoT systems [7], and we can summarize these differences in the following ways.

- **Simple-to-Complex Systems:** Typical IoT systems in deployment today are driven by energy efficient micro controllers, which typically require very long battery life (*i.e.*, 10 years and more). These are mostly driven by sensors and actuators that monitor physical environments such as building climate control or factory assembly lines, where the traffic can be characterized as being small data packets arriving intermittently with highly varying inter-arrival times, typically resulting in a low-to-average bandwidth requirements. Latency requirements vary from very stringent to orders of seconds in the context of the system they augment. In contrast, AV, MR, or DS are expected to have significant heterogeneous CPU/GPU-driven compute capabilities to be able to process the data received from the on-board 3D image sensors for computer vision purposes along with data received from other sensors for environment assessment purposes.
- **High Bandwidth Requirements:** The onboard nature of the sensors and the mobility of such systems through ground/air require high bandwidth data processing systems capable of processing high definition 3D images at a sufficiently high rate (*e.g.*, 10/30/60/100 frames/s[2] [8], [9], [10], requires the latency from image generation to inference to be bound by 100ms or less, with raw bandwidth over 4Gbps and compressed order of 100'sMbps, much higher than what current technologies can currently support, e.g. DSRC. The high bandwidth nature of these sensor applications translates to higher bandwidth requirements, especially when connected to an external network over machine-to-machine/infrastructure (M2M/M2I) interfaces to synchronize remote IoT data with the local data and make it useful for local inference.
- **Low Latency and High Reliability Requirements:** Due to the spatio-temporal nature of the data that has relevance only for a short duration into the future (*e.g.*, a few hundreds of milliseconds), while being highly location dependent, it is necessary to have sufficient bandwidth and reliability to transfer this information to the consuming points within a specific latency bound, particularly for packet streams with mission critical data.
- **Large Scale Mobility Support:** Mobile IoT systems require continuous connectivity to the edge services with network links to meet the aforementioned requirements of high bandwidth, low latency, and high reliability, and while offering such support at a much larger scale.
- **Highly Contextual Traffic:** While one of the benefits of edge compute is the contextualization of compute resources, the next generation IoT systems require the contextualization to be applied at the granularity of each device. As each of these IoT systems would be associated with high purchase/management costs, they are expected to be used to maximize its productivity.
- **Edge Compute Requirement:** Edge compute framework is indispensable to these IoT systems that should span large geography and offer ways to migrate stateful services and associated data so as to minimize the impact of end point mobility on the end-to-end application performance.

III. Transport Layer Design Choices

Considering the above requirements, a holistic solution would require an advanced system architecture with features such as new transport layer abstractions with guarantees such as deadline bound packet delivery, application aware flow prioritization, and resource reservation for mission critical flows and tasks at network and compute levels. The focus here is to propose a richer transport protocol that offers semantic data distribution capabilities rather than just byte stream data transfer, while making the system resilient to dynamic events (*i.e.*, mobility or wireless channel impairments). The transport

[2]The choice of frame rate depends on factors such as IoT system type/use, situational context and velocity.

protocol offers abstractions in the form of APIs and functionalities to applications adapting to different contexts.

Accordingly, to efficiently support edge networking within this architecture, the following functionalities becomes necessary, with the associated design choices we suggest to address it.

- **Multipoint-to-multipoint (MP2MP) Data**: In a typical AV, MR or DS scenario, data collected from a mobile UE has relevance to other systems within its proximity, as the goal of the network is to offer a global view to improve the UE's actuation decision. For this reason, the system architecture should allow for quick contextualization, discovery and dissemination of data, while conserving the energy usage by preventing UEs from processing any irrelevant information.
  $\Delta 1$: We address this point by decoupling application from network layer semantics using an *information layer* at the end-points. This information layer allows applications to address content rather than hosts, but with quick interconnection of producers' NS with distributed consumers, and offloading the resolution of NS to the transport layer rather than by the application. The data is encapsulated as stand alone named data units (NDU), with each carrying names uniquely bound to it. The data is generated by a producer as a NS, for which there may be multiple consumers. Naming schema and NDU size are chosen to optimize named stream dissemination based on reliability, networking requirements, along with enabling real time shareability. In short, the transport layer acts as a unified information layer also fostering agreement on application layer naming schemas to enable easy name discovery.
- **Service centric APIs**: To support MP2MP data distribution, the APIs offer an interface to the information layer. Furthermore, UE mobility creates additional challenges for the current host-centric transport APIs, as they overload network and host identifiers. Existing solutions that offer persistent host identifiers to applications [3] require new network infrastructure and services to be introduced to make them operational. QUIC [11] was recently proposed to handle end point mobility, with application streams mapping to persistent connection identifiers, thereby allowing session mobility. However, it is mostly optimized for client-server model rather than for MP2MP.
  $\Delta 2$: We address this requirement by using a set of information layer APIs that are exposed to applications with only identifier-based abstractions. These APIs include an ability to set secure connection with a peer. Once established, pub/sub features along with pull APIs are enabled over it, wherein data is framed as NDUs for efficient transport level multicasting.
- **Security, Privacy and Trust**: In the context of any IoT application, security, privacy and trust is a primary design consideration to prevent malicious manipulation of the end point through other IoT end points of edge network services or exposure of private data to untrusted parties. In practice, networking context drives this requirement. Hence a networking infrastructure that offers authentication, integrity and privacy functions along with supporting throughput, reliability and latency requirements becomes the desired objective. Furthermore, considering data portability, strict access control is required to properly share data within the service's trust context.
  $\Delta 3$: We address this by using a hybrid trust model. In the M2M scenario, NDUs apply security that is bound to content-centricity, while leveraging techniques that potentially would not require an Internet based certificate authority (CA) involvement for data authentication. In the M2I scenario, a transport layer integrated security model for integrigy and privacy (as in QUIC [11]), along with the functionality to authenticate request for NS from other sessions protects IoT sensor data from being shared without appropriate authorization. This allows NDUs to be subjected to privacy requirements, authenticated and encrypted over a host based secure channel.[4]
- **Dynamism**: Next-generation mobile IoT systems require uninterrupted and distributed connectivity at the service level to allow for efficient decision making using information gathered from all the relevant connected IoT systems. Therefore, it is critical to enable seamless service level mobility, and offer it in large scale. Existing solutions [12] will be highly inefficient and lack scalability, and will also require mechanisms to migrate UE state in the edge as the UE moves in the physical world to ensure that the service level latency bounds are met.
  $\Delta 4$: We address this using control plane mechanisms that include UE probing, name stream resolution function and caching at the transport session level, where the caches store named data units that are part of the stream subscribed by a peer. Here, the efficiency of recovery through caches depends on naming schema. While contextual names can help to quickly identify the missing data and to re-request them, flat names would require the applications to use manifests that are periodically inserted into NS to enable such recovery. Caches also help with service migration.[5]
- **Service Assurance**: Service assurance in terms of bandwidth, latency, and reliability has to be enforced at both compute and network levels, for robust execution of virtualized service functions (used for data aggregation, fusion, and filtering) within tight latency bounds and the need to process and deliver raw/compressed data from end points or fused data from edge points before data becomes irrevelant for actuation. This will require application-aware flow scheduling at the transport level, and marking these flows to distinguish mission critical data from the others. This application aware treatment can also be used to apply traffic prioritization at the bottleneck points, for instance at the base station.
  $\Delta 5$: Service assurance spans multiple layers of the network protocol stack and OS-level design to ensure sufficient compute, storage, and network resources are offered to flows between a UE and its ECP. Within the context of the proposed transport layer, richer end-to-end features allow efficient session and information level recovery. For instance, an application supplied latency threshold and a transport level probing mechanism can be used to determine when to migrate the edge state or to identify if the UE is in transition

[3]Host identity protocol (HIP, RFC 4423), Identifier locator network protocol (ILNP, RFC 6741), and Locator/Id separation protocol (LISP, RFC 6830).

[4]Considered hybrid security model requires minimum adaptation to the NDUs at the application level, especially when it switches them between M2M/M2I modes.

[5]As the UE connects to a different ECP, caches from the previous ECP along with connection and security context can be migrated to minimize disruption at the service level.

or connected to another point of attachment (PoA), in which case session level cache resources at the ECP can be adapted to prevent data loss.

IV. ETRA Architecture

A. Functional Components

Following novelties are emphasized through this architecture:

Design for Named Data Streaming: The architecture optimizes for quick distribution of high bandwidth sensor data from mobile IoT systems to the edge, and from edge to contextually relevant IoT devices. We evolve a secure transport-level pub/sub design carrying NDUs with scalable transport caches. We incorporate pub/sub model dissemination with name-based pull API.

Sub API is used by the consumers to subscribe to an NS, while Push API allows to securely push an NS to multiple consumers. Pull API is mainly used to recover named data after an application detects packet loss (or missing data). These APIs operate over sessions between either UE-ECP or ECP-ECP.

Data Multicasting: An ECP can be used to process data that is contextualized to its serviced UE, or it can act as an NS relay to other ECPs that can effectively aid with the creation of an overlay multicast tree.[6]

Mobility Handling: Mobility of the UE is handled using a set of control functions in the transport layer that coordinates to update the binding between the UE's new network address and the persistent session state (*i.e.* connection-ID and security context). The control functions include: (i) probing by the ECP (towards UE) to detect UE reacheability over the session lifetime; (ii) mobility event driven signaling by the UE (towards ECP), whenever it changes its PoA; (iii) re-registering the NS prefixes by a UE's transport layer's using the name stream resolution function (NSRF) [7] (to be discussed shortly); (iv) and re-resolving by ECP the UE's new PoA using its NSRF anytime during the session. These functions enable an ECP to redirect an NS to the UE at its current PoA. In addition, once a session is restored, the session cache in the ECP can be used by the UE to recover any lost data. Here the recovery is handled by the UE's consumer application.

Service Migration: Service migration here is with respect to the migration of the ECP services to a new host, which can be triggered due to mobility or lack of resource availability (*i.e.*, compute or bandwidth). This is handled by transport layer control functions that aid with the migration. With respect to migration due to mobility, control function uses the probing function mentioned earlier to ensure that the session RTT remains within the threshold as required by the application. If the transport session RTT violates this threshold, session migration is initiated. During this process: (i) a new host for the ECP is determined; (ii) session context (connection ID, security context) and the session caches are restored at the new host; (iii) services are then bootstrapped to allow the ECP application components to operate over the restored session state; (iv) migration control functions update the UE of the new network address. In addition, control functions also update the binding of the NS prefixes using the NSRF to allow the UE to re-resolve the ECP's new host during the session.

[6]The problem of interconnecting the transport endpoints of the ECPs or mobile consumer/producer end hosts for topology aware multicast is out of the scope of this paper.

[7]NSRF enables resolution of non-local NS using an external service-scoped name stream resolution system (NSRS)

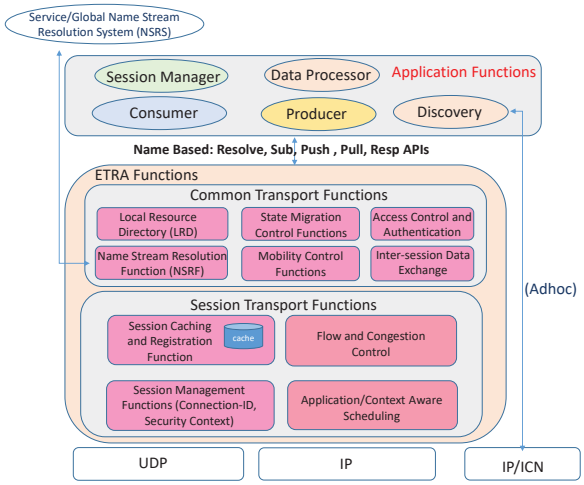

Fig. 2. Functional components of ETRA.

These functionalities are enabled using functional components shown in Figure 2. At the application level, we broadly identify four types of functional modules. While these apply to both UE and ECP, UE also includes a discovery function to contextually discover the set of nearby UEs that operate over M2M interfaces, also notified to the ECP, which can then subscribe to sensor data coming from ECPs serving those UEs. The M2M discovery functions could be contextually driven by ICN architectures [13], [14]. Accordingly, consumers can directly subscribe to the NS generated by its peers (*i.e.*, nearby UEs) from the ECP. NS name discovery by the UE depends on the role of ECP, which can be acting as a relay point or as a data processor. In the former case, names can be inferred using the naming schema design chosen by the IoT system application alone, while in the latter scenario, another round of name discovery takes place between a UE and its ECP, identifying the content based on the current binding between them. The latter scenario may necessitate content management due to service migration using, for instance, expiry policies, while at the same time this takes advantage of ECP's compute towards data contextualization to suit UE's dynamic requirements.

ETRA includes two high level transport functions: (i) session transport (ST) functions, which are required to manage a session between the end point and the edge service, and (ii) common transport (CT) functions, which are generic functions or modules to help with pub/sub, data policy management, mobility and state migration control functions.

ST functions consist of the following components.

Session Caching and Registration Management supports the use of transport layer caches, and policy associated with it during the session's lifetime. Cached data also has shareability context managed by the CT's access control function, depending on its source. For instance UE data is usable at multiple points, whereas processed data from ECP that is contextualized to a specific UE may not be shareable. The registration function allows the NS prefixes to be registered for resolution to a host address by remote consumers. The registration action is driven either by the NS policy and also used during events such as mobility or service migration.

Flow and Congestion Control should be evolved considering the data characteristics, *i.e.*, time series nature, temporal

constraints, mission critical features. These features require the use of push avoiding any data blocking with the aid of application aware scheduling, receiver driven network prioritization, and grant based flow control to handle congestion scenarios, such as used by [15] in data center scenarios, but with appropriate modification to operate in access network scenarios.

Application aware scheduling complements receiver driven flow and congestion control by using the UE based feedback on wireless connectivity to prioritize NDUs for transmissions. More specifically, the transport layer multiplexes the NS from the various sessions based on the application requirements.

CT functions consist of the following components. **Local Resource Directory** represents the local database of NS coming from local/remote producers. **Name Stream Resolution Function** (NSRF) enables resolution of non-local NS using an external name stream resolution system (NSRS). NSRS design and implementation may be influenced by application requirements. **Mobility Control Functions** help with end-to-end probing and update signaling between UE and ECP whenever network address changes. **State Migration Control Functions** include control/data migration functions between two ECPs as discussed earlier.

Access Control and Authentication manages the access policy for the local session-based caches at the host. Any ECP request to a UE's data stream requires authentication before granting access to it. **Inter-Session Exchange** helps with low-overhead based copy of data between sessions, enabling the relay function with the ECP.

B. Transport APIs

Based on the earlier discussion, Figure 3 shows the transport APIs along with their definitions, which can be classified into one of three types depending on the usage context. OpenSession and ReJoinSession are the session management APIs used by the UEs. Pub, Push and Response are the APIs used by a producer. Resolve, Sub, Pull are the APIs used by a consumer.

C. ETRA Operation

We next discuss the operation of ETRA with respect to the following phases: (i) session establishment, (ii) dynamic multicast (involving discovery), and (iii) migration (involving mobility). We illustrate the basic operation of our protocol in Figure 4. We consider a distributed application scenario corresponding to augmented vehicular reality (AVR) for autonomous vehicles [8], where a vehicle gets the view of other vehicles to generate a more detailed view of its environment. Without going into the details of AVR, our focus here is on the functionalities offered by ETRA to achieve the objectives of AVR using an edge infrastructure rather than relying solely on V2V communications (as is the case in [8]). At a high level, we assume the application components to be similar to those shown in Figure 2, with the application stack consisting of consumer, producer and data processing components. In addition, a session manager is used to bind data producing/consuming components to a trusted peer. NDUs follow the encoding of type (Push/Sub/Pull), NS identifier (NS-ID), name, and payload. We illustrate the following discussions with reference to Figure 4.

1) Session Establishment: We begin with the assumption that an AV service provider has provisioned sufficient resources at the edge for the ECP services to serve a fleet of AVs. The AVR service is then invoked based on a UE request to obtain the view of the front AV(s). Once the AV is connected to the network, an application for the AVR service discovers the closest ECP [8], after which the session managers at both ends establish a secure connection using service level authentication, thereby also creating the security context at the transport layer ($\S1$)(refer Figure 4). Consumers at either end can issue requests to discover the NS and subscribe to them, or derive the names of these NS based on a pre-existing knowledge of the naming schema by the application.[9]

Next, an end point subscribes to the NS, over which the respective producers publish data, including the shareability attributes ($\S2-4$).Published data is then pushed over the secure transport session. As the data is named contextually, transport layer can apply scheduling policies on data from multiple sessions, before handing it over the network layer. Once the data reaches the peer side, it is decrypted and sent to the consuming application for further processing.

Named data at the ECP end is saved in the cache making it available to other ECPs and to the UE (for recovery). NS-IDs are registered to a service scoped NSRS to help with discovery by the other ECPs ($\S5-6$).

2) Dynamic Multicast among ECPs: Let us consider the case of two ECP instances (ECP1 and ECP2) serving two different AVs (AV1 and AV2) that are at each other's proximity. As soon as AV1 discovers AV2 over the V2V interface ($\S7$), AV1 communicates its discovery to ECP1 over its secure data channel, after which ECP1 requests AV2's NS. Next, ECP1's consumer resolves ECP2's namespace using NSRS by invoking the host level NSRF ($\S8$). Once ECP2's namespace is resolved, ECP1's consumer starts to establish a secure session with ECP2 by sending a secure session request to ECP2 ($\S9$). Once a secure session is established between ECP1 and ECP2, ECP1's consumer initiates a request for AV2's NS ($\S10$), which is then authenticated by the ECP2's session manager to determine access to AV2's NS ($\S11$). Once the request is authenticated, it is delivered to ECP2's consumer, which then multicasts the stream over that session ($\S12-13$).

Once the data arrives over ECP1's transport session, it is cached and send to the consumer and the data processor for AV1's consumption ($\S14$). Specifically, data is first sent to ECP1's producer (which multiplexes the named data objects of multiple AVs' NS). At this point, due to the availability of AV2's NS at ECP1, this availability could be also registered to the NSRS ($\S15$).

3) Mobility and Service Migration: This is explained at a high level highlighting the utility of caches. Services leverage the session-probe primitive from the transport layer to track the UE mobility during the lifetime of a session. The API uses an application specific RTT threshold of τ to decide if an ECP service migration is required or not. Here, the need for service migration is twofold: (i) to migrate the security/connection context associated with a session (*i.e.*, host level encryption

[8]The discovery function can be handled as part of the edge platform service to interconnect a UE with the edge services.

[9]LIDAR NS from an AV1 can be named as /¡AV1-ID¿/¡Lidar¿/*, to which the ECP1's consumer can subscribe. Similarly, the processed crowdsourced data for AV1 from the ECP1 can be named as /¡AV1-ID¿/¡ECP1¿/¡AVR-view¿/*, to which the other UEs can subscribe.

API	Input	Output	Purpose
OpenSession()	ECP-IP, ECP-PORT, {Authentication}	Connection-ID, Security-Context	• Create session with ECP. • Establish security context.
RejoinSession()	ECP-IP, ECP-PORT, Old UE-IP, New-UE-IP, Conn-ID, {Security-Context}	Success or Failure	• Rejoin session after UE side PoA changes.
Pub()	Connection-ID, NS-ID, {Cache-Size}, {Access-Policy}	Success or Failure	• Open an NS channel for data generated under NS-ID with specific cache size and access policy.
Push()	Connection-ID, Named-Data, NS-ID	Success or Failure	• Push named data under NS-ID to the transport layer.
Response()	Connection-ID, Named-Data, NS-ID	Named Data	• Respond to a content request from a consumer.
Resolve()	NS-ID, {Authentication}	ECP_IP, ECP_Port	• Return IP and Port info on ECP managing data for NS-ID.
Sub()	Connection-ID, NS-ID	Success or Failure	• Make a subscription request to the ECP on a specific NS.
Pull()	Connection-ID, Named-Data, NS-ID	Named Data	• Request named data from the producer.

Fig. 3. Transport APIs.

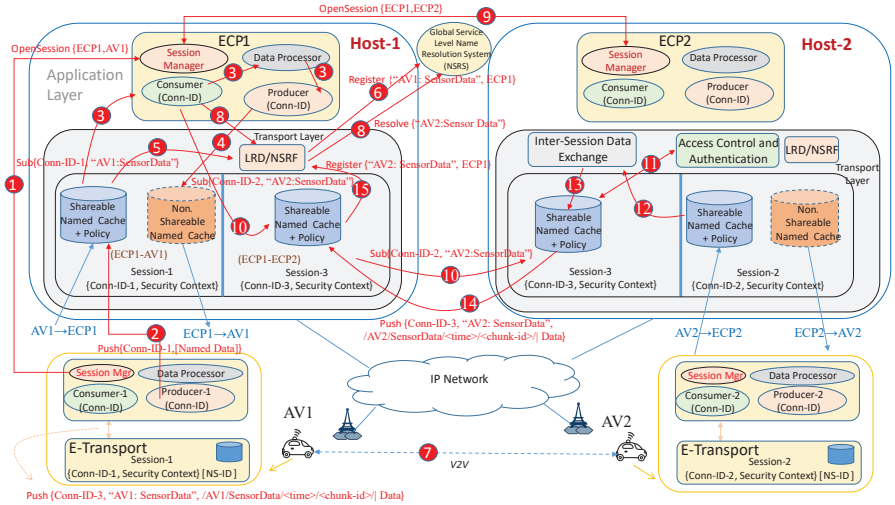

Fig. 4. Basic operation for the ETRA protocol.

and authentication keys), and (*ii*) to migrate the cache state to the new ECP. The objective here is to allow the UE connected to the new PoA to immediately start sending/receiving data without further session negotiations.

Service migration is triggered once the ECP determines that the RTT threshold is violated (for a certain duration), which is primarily driven by the UE mobility. If, after a handover, UE-ECP connection continues to satisfy the RTT threshold, then the same ECP continues to serve the UE. In this case, the session cache in the ECP can help applications to recover data lost during the transition. Otherwise, the current ECP uses an operator driven network service to identify the next closest edge with sufficient compute resources to host the ECP workload associated with UE, after which the required service containers are orchestrated. Next, the transport session state is transferred securely so that the future data from the UE can be properly authenticated and decrypted before handing it to the ECP. Once the security context is transferred, consumer and producer caches are transferred to the new ECP.

After migration, UE resolves the new ECP. Once the UE learns the new ECP's IP, it continues to use the same session context (*i.e.*, connection-ID and shared encryption key) between consumer and producer of the UE/ECP.[10]

V. Conclusion

Even though next generation IoT systems, such as autonomous vehicles, drones, and mobile robots, present economic benefits and opportunities, the question remains as to whether the current IP-based Internet architecture can meet their challenges including realtime data sharing, mobility and security. Large scale operation of these IoT systems require an intelligent edge at all levels offering service-level load distribution and new transport layer functionalities to support mission critical data streams. However, integrating these systems safely into human operating conditions using only device-driven guidance system has been a challenge, especially with not enough consideration paid to the role networks can play in enhancing their operational efficiency.

This paper takes an important step towards understanding the requirements of these applications and the possible direction in adapting the existing IP network architecture to meet these needs. We suggest the use of an enhanced transport layer that supports named streams of data that are mediated through edge computing points, which are part of the infrastructure. We propose an information layer that decouples the application from the underlying network layer semantics, and describe the features of a transport layer architecture that supports rapid data sharing, mobility, service migration while providing security and privacy.

[10]After the new association the NS would be renamed to / <AV1-ID> / <ECP2> / <AVR-view> /∗.

References

[1] "AV System Safety, https://www.youtube.com/watch?v=a7L51u23YoM."

[2] R. Zhang, "UAV meets wireless communication in 5G and beyond: Main research challenges and key enabling techniques," IEEE WCNC Tutorial, 2018.

[3] R. Jain, "Unmanned aerial systems: Networking applications, challenges and issuess," Midwest Drone Introduction, Keynote, 2016.

[4] S. P. Chinchali *et al.*, "Neural networks meet physical networks: Distributed inference between edge devices and the cloud," in *Proceedings of the ACM HotNets'18*, 2018.

[5] "Boston Dynamics, https://www.bostondynamics.com."

[6] B. Zhang *et al.*, "The cloud is not enough: Saving IoT from the cloud," in *Proceedings of USENIX HotCloud*, 2015.

[7] S. H. Shah *et al.*, "A survey: Internet of Things (IOT) technologies, applications and challenges," in *IEEE SEGE*, 2016.

[8] H. Qiu *et al.*, "AVR: Augmented vehicular reality," in *Proceedings of ACM MobiSys*, 2018.

[9] S.-C. Lin *et al.*, "The architectural implications of autonomous driving: Constraints and acceleration," in *Proceedings of the ACM ASPLOS*, 2018.

[10] L. Liu *et al.*, "Edge assisted real-time object detection for mobile augmented reality," in *ACM Mobicom (first round preprint)*, 2019.

[11] A. Langley *et al.*, "The QUIC transport protocol: Design and internet-scale deployment," in *Proceedings of the ACM SIGCOMM*, 2017.

[12] F. Giust *et al.*, "MEC deployments in 4G and evolution towards 5G," ETSI White Paper, Tech. Rep., 2018.

[13] V. Jacobson *et al.*, "Networking named content," in *Proceedings of the ACM CoNEXT*, 2009.

[14] G. Grassi *et al.*, "Poster: Vehicular inter-networking via named data," *Proceedings of the ACM HotMobile*, 2013.

[15] B. Montazeri *et al.*, "Homa: A receiver-driven low-latency transport protocol using network priorities," in *Proceedings of the ACM SIGCOMM*, 2018.