Network Working Group
Internet-Draft
Intended status: Informational
Expires: 11 January 2021

X. de Foy
A. Rahman
InterDigital Communications, LLC

10 July 2020

Impact of Mobility on Discovery in COIN
draft-defoy-coinrg-mobile-discovery-00

Abstract

Service, data and resource discovery is an important aspect of computing in the network. While this aspect has been studied, including in COINRG, this document looks more specifically at the influence of mobile devices on COIN discovery. Related research challenges are described and discussed.

Table of Contents

1. Introduction

Discovery in COIN relates to edge services including computing resources, computing services and data. In this document, we focus on the influence of mobile devices on discovery in COIN, both when mobile devices are consuming or producing edge computing services. We use the following terms:

* Edge computing service: a general concept including offering computing and storage resources to other devices or to a platform, through an API that enables allocating computing/storage resources, onboarding a program, running a program; offering a computing service such as an API to a software program running on the device; or offering a data service such as a data stream or an API to access data generated by, or stored on, the device.

* Edge computing service provider: a device or platform providing such a service. We especially consider cases where a mobile device acts as a provider.

* Edge computing service consumer: a (possibly mobile) device discovering, requesting and obtaining access to such a service.

* Network provider: an entity providing network connectivity to the mobile devices discussed in this draft. It can be a 5G network operator, or an enterprise or home network operator.

This document is related, and aims to be complementary, to [I-D.mcbride-edge-data-discovery-overview], which studies data discovery in COIN environments.

Service and resource discovery has been studied more generally for distributed edge computing. For example [Varghese] identifies challenges including scaling, support for heterogeneous environments, support for real-time benchmarking. As examples of system designs, [Gedeon] describes a distributed brokering system for discovery, and [Mastorakis] describes an ICN-based discovery scheme.

2. Challenges

Due to the mobility of service/data/resource providers and consumers, service discovery is typically used more often when involving mobile devices (i.e. not only during initial service setup, but continuously during service operation), and failures can lead to less stable services. Mobility also brings specific challenges to service discovery in edge computing, in term of scalability, support for multiple and frequently changing network interfaces, and service continuity.

2.1. Scalability

2.1.1. Challenge Description

From its distributed nature, edge computing generally improves scalibility of services/data/resources. However, this puts more demand on mobile networks. Scalability is a concern with discovery involving wireless mobile devices, especially because of the scarce nature of the wireless medium: mobile devices should use as little resources as practical to determine whether a service or resource is present, or to advertise their own service/resource. Additionally, multicast over wireless is also expensive, as described in section 2.2 of [RFC7558]. Moreover, even beyond the first wireless hop, dense deployments of mobile devices can result in high churn, which may generate an unwanted constant traffic activity for edge computing related service discovery in edge networks.

2.1.2. Discussion

A common strategy to increase service discovery scalability for wireless devices is to use pre-attachment or pre-connection discovery methods, as an initial stage for service discovery. These methods provide information to the mobile device, to help narrow down the number of access points or data networks that are eligible to access the service.

* One example is 802.11aq [IEEE-802.11aq], where hashes or bloom filters summarizing service names can be advertised by access points, prior to attachment; the mobile device can also send a service-specific request through the access point, also prior to attachment. 802.11aq uses service names as defined in [RFC6335].

* A 5G mobile device will be able to send a request to an application function (Edge Enabler Server), including parameters such as application client ID, requested response time, bandwidth, compute, memory and storage resources. The reply will include information on the selected edge application server [_3GPP.23.558].

As illustrated in multiple edge computing system designs ([Kaur], [_3GPP.23.558], [Gedeon]), as part of the discovery process a network node may collect QoS requirements from the service consumer, and then select or reserve resources for this service. One challenge may be to limit the impact of this step on edge computing discovery. Collecting and using requirements may for example be performed by the service consumer, when used with passive discovery methods that provide enough information. Collecting and using requirements may also occur at different stages of discovery (e.g., pre-connection, or later after connecting to an edge computing platform). Finally, in some cases this step may not be needed at all (e.g., for a best effort edge computing service, or if a QoS is implied for a given service).

2.2. Multiple Interfaces and Data Networks

2.2.1. Challenge Description

Mobile devices can have multiple radios, resulting in the added challenge of determining which network interface(s) to use for discovery, either initially or for session continuity when relocating. Additionally, even once a mobile device is attached to an access point, multiple (local- and wide-area) data network may be locally accessible, also resulting in multiple network interfaces on the device. The problem of dealing with multiple interfaces is not unique to mobile device (e.g. routers participating in edge computing will have similar issues), however with mobile devices network interfaces are much more dynamic as part of normal operation. This problem area has been discussed in MIF [I-D.cao-mif-srv-dis-ps], however not within the context of edge computing.

2.2.2. Discussion

One strategy is to connect to several available access points and discover service instances concurrently, following a happy eyeball strategy [RFC8305]. Otherwise, a mobile device should select the network interfaces to use based on available information (including from pre-attachment/pre-connection methods above). Once a connection is established, multiple discovery methods are available:

* In passive discovery methods network nodes advertise information to end devices, without requiring a specific request. One possibility is to extend existing passive discovery methods for edge computing services. For example:
  - Leveraging provisioning domains (e.g., listing available service and instance names),
  - Leveraging router advertisements (e.g., advertising Virtual Infrastructure Management resources, as described in [I-D.bernardos-sfc-fog-ran])
  - Leveraging DHCP signalling (e.g., advertising the IP address of an edge computing platform server).
* Active discovery protocols include DNS-based discovery methods such as DNS-SD [RFC6763] and mDNS [RFC6762]. Mobile service producers can make themselves known using multicast (for mDNS) or register with the DNS system, e.g. using [I-D.ietf-dnssd-srp]

One additional challenge is to make multi-interface discovery methods available as early as possible to save resources (e.g. pre-attachment/connection). For example, a recent proposal in 5G is to use policy information to deploy provisioning domains on mobile devices prior to establishing a PDU connection: the mobile device can then look at provisioning domain attributes and determine which data network to use and related connection parameters [_3GPP.23.748].

2.3. Service Continuity

2.3.1. Challenge Description

Service continuity and latency can also be impacted by service or resource discovery. When a mobile device (either service consumer or provider) moves to a new location, a new service instance may need to be discovered to maintain the level of service (e.g. keep rendering or processing video without losing a frame or more than _n_ frames). In cases where edge computing is used for real time applications with stringent requirements, the time used to discover a new edge computing instance influences the level of service.

2.3.2. Discussion

Mobile devices are faced with the challenge to select a proper service continuity strategy, each time a new access point becomes available or unavailable. Edge computing service discovery methods may need to provide information not only to facilitate this selection, but to factor in service continuity strategies within the discovery process. Typical edge computing service continuity strategies are: a mobile device may keep connecting to the same serving instance through a new AP (using connection migration or multiple paths); or a mobile device may discover a new instance and then use it to replace or complement its connection to the first instance. For example, a mobile device may be in range of 2 APs, one suitable for the first strategy and the other one suitable for the second.

3. Security Considerations

One concern is for the consumer to trust that discovery information relayed by the network provider is legitimate, to avoid, for example, phishing or denial of service attacks. Another concern is for the provider to limit the amount of information given to unauthenticated requesters. For pre-connection discovery, these types of concerns are typically addressed by authorization (e.g. in 5G, a device must be attached to the network prior to discover services) or hashing (e.g. in 802.11aq service names are advertised through hashes or bloom filters).

4. Acknowledgment

The authors would like to thank Chonggang Wang for his valuable comments and suggestions on this document.

5. Informative References

[Gedeon] Gedeon, J., Meurisch, C., Bhat, D., Stein, M., Wang, L., and M. Muhlhauser, "Router-Based Brokering for Surrogate Discovery in Edge Computing", IEEE 37th International Conference on Distributed Computing Systems Workshops (ICDCSW) , 2017, <https://ieeexplore.ieee.org/abstract/document/7979808>.

[I-D.bernardos-sfc-fog-ran] Bernardos, C., Rahman, A., and A. Mourad, "Service Function Chaining Use Cases in Fog RAN", Work in Progress, Internet-Draft, draft-bernardos-sfc-fog-ran-07, 11 March 2020, <http://www.ietf.org/internet-drafts/draft-bernardos-sfc-fog-ran-07.txt>.

[I-D.cao-mif-srv-dis-ps] Cao, Z. and A. Ding, "Service Discovery in a Multiple Connection Environment: Problem Statement", Work in Progress, Internet-Draft, draft-cao-mif-srv-dis-ps-03, 27 August 2013, <http://www.ietf.org/internet-drafts/draft-cao-mif-srv-dis-ps-03.txt>.

[I-D.ietf-dnssd-srp] Cheshire, S. and T. Lemon, "Service Registration Protocol for DNS-Based Service Discovery", Work in Progress, Internet-Draft, draft-ietf-dnssd-srp-02, 8 July 2019, <http://www.ietf.org/internet-drafts/draft-ietf-dnssd-srp-02.txt>.

[I-D.mcbride-edge-data-discovery-overview] McBride, M., Kutscher, D., Schooler, E., and C. Bernardos, "Edge Data Discovery for COIN", Work in Progress, Internet-Draft, draft-mcbride-edge-data-discovery-overview-03, 29 January 2020, <http://www.ietf.org/internet-drafts/draft-mcbride-edge-data-discovery-overview-03.txt>.

[IEEE-802.11aq] IEEE, ., "IEEE 802.11 Specifications Amendment 5: Preassociation Discovery", IEEE Std 802.11aq-2018 , 2018.

[Kaur] Kaur, K., Dhand, T., Kumar, N., and S. Zeadally, "Container-as-a-service at the edge: Trade-off between energy efficiency and service availability at fog nano data centers", IEEE wireless communications , 2017, <https://ieeexplore.ieee.org/document/7955911>.

[Mastorakis]
Mastorakis, S. and A. Mtibaa, "Towards Service Discovery and Invocation in Data-Centric Edge Networks", IEEE 27th International Conference on Network Protocols (ICNP) , 2019, <https://ieeexplore.ieee.org/abstract/document/8888081>.

[RFC6335] Cotton, M., Eggert, L., Touch, J., Westerlund, M., and S. Cheshire, "Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry", BCP 165, RFC 6335, DOI 10.17487/RFC6335, August 2011, <https://www.rfc-editor.org/info/rfc6335>.

[RFC6762] Cheshire, S. and M. Krochmal, "Multicast DNS", RFC 6762, DOI 10.17487/RFC6762, February 2013, <https://www.rfc-editor.org/info/rfc6762>.

[RFC6763] Cheshire, S. and M. Krochmal, "DNS-Based Service Discovery", RFC 6763, DOI 10.17487/RFC6763, February 2013, <https://www.rfc-editor.org/info/rfc6763>.

[RFC7558] Lynn, K., Cheshire, S., Blanchet, M., and D. Migault, "Requirements for Scalable DNS-Based Service Discovery (DNS-SD) / Multicast DNS (mDNS) Extensions", RFC 7558, DOI 10.17487/RFC7558, July 2015, <https://www.rfc-editor.org/info/rfc7558>.

[RFC8305] Schinazi, D. and T. Pauly, "Happy Eyeballs Version 2: Better Connectivity Using Concurrency", RFC 8305, DOI 10.17487/RFC8305, December 2017, <https://www.rfc-editor.org/info/rfc8305>.

[Varghese] Varghese, B., Wang, N., Barbhuiya, S., Kilpatrick, P., and D.S. Nikolopoulos, "Challenges and Opportunities in Edge Computing", IEEE International Conference on Smart Cloud , 2016, <https://ieeexplore.ieee.org/document/7796149>.

[_3GPP.23.558]
3GPP, ., "Architecture for enabling Edge Applications; (Release 17)", 3GPP TS 23.558 , 2020, <http://www.3gpp.org/ftp/Specs/html-info/23558.htm>.

[_3GPP.23.748]
3GPP, ., "Study on enhancement of support for Edge Computing in 5G Core network (5GC)", 3GPP TS 23.748 , 2020, <http://www.3gpp.org/ftp/Specs/html-info/23748.htm>.

Authors' Addresses

Xavier de Foy
InterDigital Communications, LLC
1000 Sherbrooke West
Montreal H3A 3G4
Canada

Email: xavier.defoy@interdigital.com

Akbar Rahman
InterDigital Communications, LLC
1000 Sherbrooke West
Montreal H3A 3G4
Canada

Email: akbar.rahman@interdigital.com

COINRG
Internet-Draft
Intended status: Informational
Expires: September 10, 2020

I. Fink
K. Wehrle
RWTH Aachen University

March 9, 2020

Enhancing Security and Privacy with In-Network Computing
draft-fink-coin-sec-priv-00

Abstract

With the growing interconnection of devices, cyber-security and data protection are of increasing importance. This is especially the case regarding cyber-physical systems due to their close entanglement with the physical world. Misbehavior and information leakage can lead to financial and physical damage and endanger human lives and well-being. Thus, hard security and privacy requirements are necessary to be met. Furthermore, thorough investigation of incidents is essential for ultimate protection. In-network computing allows the processing of traffic and data directly in the network and at line-rate. Thus, the in-network computing paradigm presents a promising solution for efficiently providing security and privacy mechanisms as well as event analysis. This document discusses select mechanisms to demonstrate how in-network computing concepts can be applied to encounter current shortcomings of cyber-security and data privacy.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 10, 2020.

Copyright Notice

Table of Contents

1. Introduction

Several deficiencies emerge from cyber-physical systems (CPS) such as the (Industrial) Internet of Things (IoT). Everyday things are equipped with sensors and CPUs to allow for automatization and make life more comfortable. The deployment of additional sensors supports the processing efficiency in Industrial Control Systems (ICS). The entanglement of the sensors with the physical world leads to a high sensitivity of the transmitted and collected data. At the same time, devices are increasingly connected to the Internet to enable, e.g., processing of data on cloud servers or exchange with other systems.

Devices in CPS are often resource-constrained and do not offer the possibility to implement elaborate security mechanisms. Furthermore, legacy devices and communication protocols are often still in use in industrial networks but were not designed to face the security and privacy challenges the new interconnection brings. Thus, communication and access are often unprotected, providing new attack surfaces with severe consequences: leakage of private data endangers the users' privacy. The leakage of business secrets bears the risk

of severe financial damage. Manipulation of ICSs can lead to downtimes in the best case or, financially worse, faulty production results. Last, the failure of CPS can lead to personal injury or even death. As a consequence of the described risks, we need security and privacy measures tailored to the new situation. Upgrading legacy devices with protection mechanisms is an effortful and expensive procedure. A promising approach for retrofitting security nevertheless is the deployment of suitable mechanisms within the network. To date, this is mainly realized using middle-boxes, leading to overhead and need for additional hardware.

While proper prevention and detection of attacks in the (Industrial) IoT is an unresolved issue, the after-treatment of incidents in networks offers room for improvement in general. We can use network forensics to retrace and comprehend the origin and course of malicious events. However, realization requires high effort and costs in traditional networks.

The common problem of all shortcomings is that traditional networking devices only allow for fixed-function deployment. Software-defined networking (SDN) enables more flexible traffic handling in the network by separating control and data plane. However, the use of fixed-function switches still restricts primary approaches like OpenFlow. Those switches match traffic against a fixed set of protocol headers to decide if and where it should be forwarded. Furthermore, consultation of the remote control plane leads to communication overhead and delays, which is especially unfavorable in the context of time-sensitive applications, e.g., in industry.

INC, in contrast, covers the shortfalls of traditional networks and SDN by allowing actual programming of the switches. This programmability leads to dynamic and custom processing of network packets at line-rate. Thus, security-related functions and packet inspection can be implemented and applied right at the switch.

This draft explores the opportunities of INC for improving security and privacy as follows: we first describe feasible mechanisms for preventing attacks and intrusion in the first place. Then, we present which mechanisms we can implement with INC for detecting intrusion and undesired behavior when it has already taken place. Last, we explore how INC can improve network forensics for analyzing and following up incidents, which helps to prevent future attacks.

2. Protection Mechanisms

The common ground for providing security and data privacy is to protect against unauthorized access. That protection is primarily provided by deploying the basic security mechanisms encryption,

integrity checking, authentication, and authorization. Those are especially often missing in resource-constrained environments. [RFC7744] thoroughly discusses the need for authentication and authorization in resource-restrained environments. [RFC8576] presents security and privacy risks and challenges specific to the IoT. In the following, we describe how INC can help to retrofit suitable mechanisms.

2.1. Encryption and Integrity Checking

Encryption is critical to preserve confidentiality when transmitting data. Integrity checks prevent undetected manipulation, which can remain unnoticed even despite encryption, e.g., in case of flipped bits. Due to resource-constraints, many devices in CPS do not provide encryption or calculation of check-sums.

Complex cryptography is not supported by current programmable switches either. However, this might change in the future, which would allow retrofitting encryption and integrity checks at networking devices. Concretely, using INC with suitable hardware, data could be encrypted and supplemented with a check-sum directly at the first networking device passed by the respective data packet. The packet is then forwarded through the network or Internet to its designated destination. Decryption and integrity checks can be executed at the last networking device before the destination. Alternatively, this can be implemented at the destination if supported by the respective device. This approach does not require deployment or forwarding to additional middleboxes. Thus, no additional attack surface or processing overhead is introduced, which is essential for time-sensitive processes as often at hand in industry.

Overall, INC has the potential to help maintain confidentiality and integrity efficiently, and thus the availability of resource-constrained or legacy devices. Questions to clarify are if and at which costs hardware for enabling cryptographic calculations could and should be embedded in future generations of programmable networking devices.

2.2. Authorization and Authentication

Authorization and authentication mechanisms are needed to avoid unauthorized access to devices and their manipulation in the first place. With INC, networking devices can flexibly decide whether to forward packets, thus enforce authorization and authentication checks.

One possibility for authorization is to conduct a handshake between the sender and networking device before starting the communication with the industrial device. If not feasible in switch hardware, the respective calculations can be conducted in the control plane. In the case of success, the sender is added to a list of authorized communication partners. The decision is then enforced by the switch. Since authorization is only needed when starting or refreshing a connection, the necessity and overhead for consulting the control plane are limited.

The sender can append a secret token for authentication to packets directed to an industrial device. Then, the last networking device can authenticate the sender and forward the actual data only in case of success and drop it otherwise. One possibility to avoid eavesdropping of the token is the use of hash chains. Secure reinitialization can again be done using the control plane, which usually has the resources for conducting encrypted communication.

In the case of unsuccessful authorization or authentication, networking devices can inform the network administrator about possible intrusion of the system.

Undesired traffic can emerge even from authorized and authenticated devices. A solution is to add policy-based access control, on which we elaborate in the next subsection.

2.3. Behavioral and Enterprise Policies

Control processes can include communication between various parties. Even despite authorization and authentication mechanisms, undesired behavior can occur. For instance, malicious third-party software might be installed at the approved device. Regarding communication between two legacy devices, authentication might not be possible at all. An effective way to exclude malicious behavior nevertheless is policy-based access control.

[RFC8520] proposes the Manufacturer Usage Description (MUD), a standard for defining the communication behavior of IoT devices, which use specific communication patterns. The definition is primarily based on domain names, ports, and use of protocols (e.g., TCP and UDP). Further characteristics as the TLS usage [I-D.draft-reddy-opsawg-mud-tls-03] or the required bandwidth of a device [I-D.draft-lear-opsawg-mud-bw-profile-01] can help to define connections more narrowly.

By defining the typical behavior, we can exclude deviating communication, including undesired behavior. Likewise to IoT devices, industrial devices usually serve a specific purpose. Thus,

the application of MUD or similar policies is possible in industrial scenarios as well.

The problem that remains to date is the efficient enforcement of such policies in the form of fine-granular and flexible traffic filtering. While middle-boxes increase costs and processing overhead, primary SDN approaches as OpenFlow allow only filtering based on match-action rules regarding fixed protocol header fields. Evaluation of traffic statistics for, e.g., limiting the bandwidth, requires consultation of the remote controller. This leads to latency overheads, which are not acceptable in time-sensitive scenarios.

In contrast, the INC paradigm allows flexible filtering even concerning the content of packets and connection metadata. Furthermore, traffic filtering can be executed at line-rate in the switch.

Going one step further, not only network communication behavior of devices can be defined in policies. As [KANG] shows, INC can be used to consider additional (contextual) parameters, e.g., the time of day or activity of other devices in the network. Furthermore, companies can define advanced policies to, e.g., authorize specific users or subnets.

Additional protection of data is needed to preserve business secrets and the privacy of individuals. We show in the following subsection how INC can contribute to data anonymization.

2.4. Anonymization

Due to its interconnection with the physical world, the generation of sensitive data is inherent to CPS. Smart infrastructure leads to the collection of sensitive user data. In industrial networks, information about confidential processes is gathered. Such data is increasingly shared with other entities to increase production efficiency or enable automatic processing.

Despite the benefits of data exchange, manufacturers, as well as individuals, might not want to share sensitive information. Again, deployment of privacy mechanisms is usually not possible at resource-constrained or legacy devices. INC has the potential to flexibly apply privacy mechanisms at line-rate.

Data can be pseudonymized at networking devices by, e.g., extracting and replacing specific values. Furthermore, elaborate anonymization techniques can be implemented in the network by sensibly decreasing the data accuracy. For example, concepts like k-Anonymity can be applied by aggregating the values of multiple packets before

forwarding the result. Noise addition can be implemented by adding a random number to values. Similarly, the state-of-the-art technique differential privacy can be implemented by adding noise to responses to statistical requests.

Even though the INC paradigm shows the potential to deploy described privacy mechanisms within the network, research is needed to clarify the feasibility of the proposed concepts.

3. Intrusion and Anomaly Detection

Ideally, attacks are prevented from the outset. However, in the case of incidents, fast detection is critical for limiting damage. Deployment of sensors, e.g., in industrial control systems, can help to monitor the system state and detect anomalies. This can be used in combination with INC to detect intrusion as well as to provide advanced safety measures, as described in the following.

3.1. Intrusion Detection

Data of sensors or communication behavior can be compared against expected patterns to detect intrusion. Even if intrusion prevention is deployed and connections are allowed when taken individually, subtle attacks might still be possible. In this case, e.g., a series of values might be again out of line if considered at large. Anomaly detection can be used to detect such abnormalities and notify the network administrator for further assessment.

While anomaly detection is usually outsourced to middleboxes or external servers, INC provides the possibility to detect anomalies at-line rate, e.g., by maintaining statistics about traffic flows. This decreases costs and latency, which is valuable for a prompt reaction.

Besides intrusion, anomalies can also imply safety risks. In the following, we pick up the potential of INC to support safety.

3.2. Dead man's switch

[I-D.draft-kunze-coin-industrial-use-cases-01] addresses the potential of INC for improving industrial safety. Detection of an anomaly in the sensor data or operational flow can be used to automatically trigger an emergency shutdown of a system or single components if the data indicates an actual hazard. Apart from that, other safety measures like warning systems or isolation of areas can be implemented. While we do not aim at replacing traditional dead man's switches, we see the potential of INC to accelerate the

detection of failures. Thus, INC can valuably complement existing safety measures.

4. Incident Reappraisal

After the detection and treatment of an incident, it is important to conduct Network Forensics to investigate the origin and spreading of the related activity. The results of this analysis can be used to adapt protection mechanisms and prevent similar events in the future. For enabling potential investigation, traffic records are constantly collected for each flow in a network. This requires additional hardware in traditional networks. Furthermore, it might be preferable to exclude, e.g., specific subnets from the analysis. This is not possible with traditional networking devices, leading to storage and processing overhead.

With INC, flow records can be created directly at the switch when forwarding a packet. Furthermore, record generation can be done more flexibly, e.g., by applying fine-granular traffic filtering. Also, header fields of particular interest can be efficiently extracted. Therefore, INC can considerably decrease the load and increase the efficiency of network forensics. This leads in turn to a better understanding of attacks and security.

5. Security Considerations

When implementing security and privacy measures in networking devices, security and failure resistance of the networking devices themselves is critical. Related research questions to clarify in the future are stated in [I-D.draft-kutscher-coinrg-dir-01].

6. IANA Considerations

N/A

7. Conclusion

In-network computing has the potential to improve and retrofit security and privacy, especially in concern of resource-restrained and legacy devices.

First, INC can provide intrusion prevention mechanisms like authentication and efficient enforcement of (context-based) policies. Encryption and integrity checks are limited by the current hardware but might be realizable in the future.

Second, INC allows examining packet contents at networking devices, which can be used to implement fast anomaly and intrusion detection in the network.

Last, INC can contribute to an efficient and targeted incident analysis.

Investigation of the feasibility of the presented mechanisms is subject to future research.

8. Informative References

[I-D.draft-kunze-coin-industrial-use-cases-01]
Kunze, I. and K. Wehrle, "Industrial Use Cases for In-Network Computing", draft-kunze-coin-industrial-use-cases-01 (work in progress), November 2019.

[I-D.draft-kutscher-coinrg-dir-01]
Kutscher, D., Karkkainen, T., and J. Ott, "Directions for Computing in the Network", draft-kutscher-coinrg-dir-01 (work in progress), November 2019.

[I-D.draft-lear-opsawg-mud-bw-profile-01]
Lear, E. and O. Friel, "Bandwidth Profiling Extensions for MUD", draft-lear-opsawg-mud-bw-profile-01 (work in progress), July 2019.

[I-D.draft-reddy-opsawg-mud-tls-03]
Reddy, T., Wing, D., and B. Anderson, "MUD (D)TLS profiles for IoT devices", draft-reddy-opsawg-mud-tls-03 (work in progress), January 2019.

[KANG] Kang, Q., Morrison, A., Tang, Y., Chen, A., and X. Luo, "Programmable In-Network Security for Context-aware BYOD Policies", In Proceedings of the 29th USENIX Security Symposium (USENIX Security 20), August 2020, <https://www.usenix.org/conference/usenixsecurity20/presentation/kang>.

[RFC7744] Seitz, L., Ed., Gerdes, S., Ed., Selander, G., Mani, M., and S. Kumar, "Use Cases for Authentication and Authorization in Constrained Environments", RFC 7744, DOI 10.17487/RFC7744, January 2016, <https://www.rfc-editor.org/info/rfc7744>.

[RFC8520] Lear, E., Droms, R., and D. Romascanu, "Manufacturer Usage Description Specification", RFC 8520, DOI 10.17487/RFC8520, March 2019, <https://www.rfc-editor.org/info/rfc8520>.

[RFC8576] Garcia-Morchon, O., Kumar, S., and M. Sethi, "Internet of Things (IoT) Security: State of the Art and Challenges", RFC 8576, DOI 10.17487/RFC8576, April 2019, <https://www.rfc-editor.org/info/rfc8576>.

Authors' Addresses

Ina Berenice Fink
RWTH Aachen University
Ahornstr. 55
Aachen D-52062
Germany

Phone: +49-241-80-21419
Email: fink@comsys.rwth-aachen.de

Klaus Wehrle
RWTH Aachen University
Ahornstr. 55
Aachen D-52062
Germany

Phone: +49-241-80-21401
Email: wehrle@comsys.rwth-aachen.de

COINRG
Internet-Draft
Intended status: Informational
Expires: September 10, 2020

I. Kunze
K. Wehrle
RWTH Aachen University

March 09, 2020

Industrial Use Cases for In-Network Computing
draft-kunze-coin-industrial-use-cases-02

Abstract

Cyber-physical systems and the Industrial Internet of Things are characterized by diverse sets of requirements which can hardly be satisfied using standard networking technology. One example are latency-critical computations which become increasingly complex and are consequently outsourced to more powerful cloud platforms for feasibility reasons. The intrinsic physical propagation delay to these remote sites can already be too high for given requirements. The challenge is to develop techniques that bring together these requirements. Utilizing available computational capabilities within the network for in-network computing concepts can be a solution to this challenge. This document discusses selected industrial use cases to demonstrate how in-network computing concepts can be applied to the industrial domain and to point out essential requirements of industrial applications.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 10, 2020.

Copyright Notice

Table of Contents

1. Introduction

The Internet is based on a best-effort network that provides limited guarantees regarding the timely and successful transmission of packets. This design-choice is suitable for general Internet-based applications, but specialized industrial applications demand a number of strict performance guarantees, e.g., regarding real-time capabilities, which cannot be provided over regular best-effort networks.

Enhancements to the standard Ethernet such as Time-Sensitive-Networking [TSN] try to achieve the requirements on the link layer by statically reserving shares of the bandwidth. These concepts are well-suited for industrial settings with well understood communication patterns where the communication paths are encapsulated at the factory sites. In the Industrial Internet of Things (IIoT), however, more and more parts of the industrial production domain are interconnected. This increases the complexity of the industrial

networks, makes them more dynamic, and creates more diverse sets of requirements. Furthermore, process control is imagined to be exercised from remote clouds for feasibility reasons which is why solutions on the link layer alone are not sufficient in these scenarios.

Common components of the IIoT can be divided into three categories as illustrated in Figure 1. Following [I-D.mcbride-edge-data-discovery-overview], EDGE DEVICES, such as sensors and actuators, constitute the boundary between the physical and digital world. They communicate the current state of the physical world to the digital world by transmitting sensor data or let the digital world interact with the physical world by executing actions after receiving (simple) control information. The processing of the sensor data and the creation of the control information is done on COMPUTING DEVICES. They range from small-powered controllers close to the EDGE DEVICES, to more powerful edge or remote clouds in larger distances. The connection between the EDGE and COMPUTING DEVICES is established by NETWORKING DEVICES. In the industrial domain, they range from standard devices, e.g., typical Ethernet switches, which can interconnect all Ethernet-capable hosts, to proprietary equipment with proprietary protocols only supporting hosts of specific vendors.

The challenge is to develop concepts that can include off-premise entities (such as distant cloud platforms) as well as proprietary

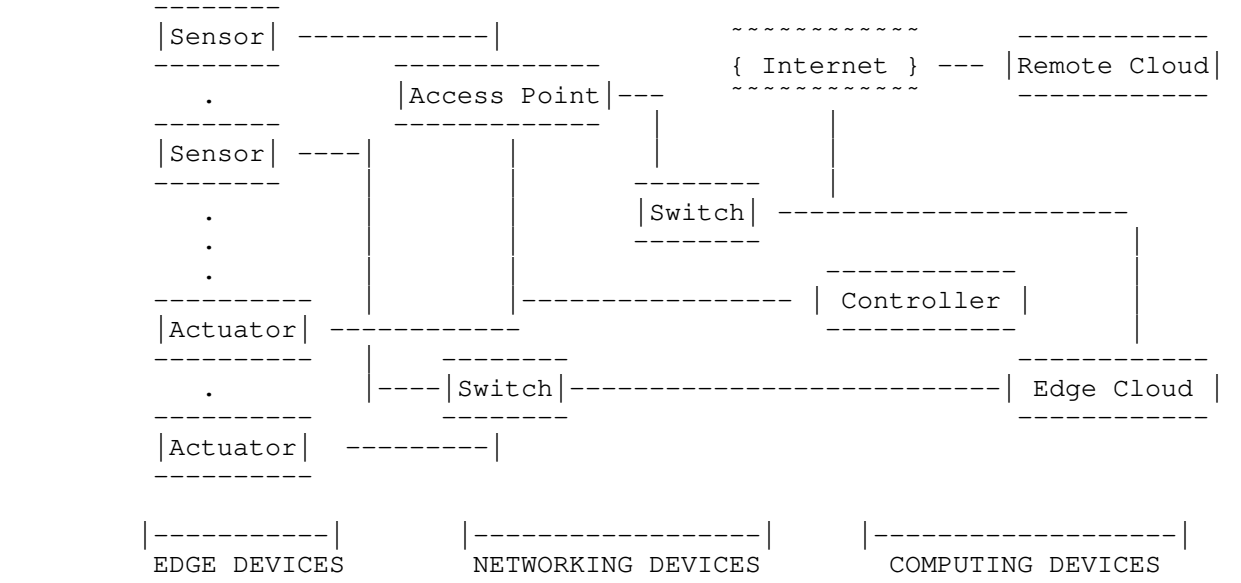

Figure 1: Industrial networks show a high level of heterogeneity.

hosts into the communication and still satisfy the performance requirements of modern industrial networks. The in-network computing paradigm presents a promising starting point because (pre-)processing data within the network can speed up the communication, e.g., by reducing the amount of transmitted data and thus congestion. Flexibly distributing the computation tasks across the network helps to manage dynamic changes. Specifying general requirements for the different application scenarios is difficult due to the mentioned diversity. This draft characterizes and analyzes three distinct scenarios to showcase potential requirements for the industrial production domain and to illustrate how in-network computations can be helpful.

2. In-Network Control / Time-sensitive applications

The control of physical processes and components of a production line is essential for the growing automation of production and ideally allows for a consistent quality level. Traditionally, the control has been exercised by control software running on programmable logic controllers (PLCs) located directly next to the controlled process or component. This approach is best-suited for settings with a simple model that is focussed on a single or few controlled components.

Modern production lines and shop floors are characterized by an increasing amount of involved devices and sensors, a growing level of dependency between the different components, and more complex control models. A centralized control is desirable to manage the large amount of available information which often has to be pre-processed or aggregated with other information before it can be used. PLCs are not designed for this array of tasks and computations could theoretically be moved to more powerful devices. These devices are no longer close to the controlled objects and induce additional latency.

It is worthwhile to investigate whether the outsourcing of control functionality to distant computation platforms is viable because these platforms have a high level of flexibility and scalability. In the following, we describe the requirements and characteristics of the control setting in more detail.

2.1. Characterization and Requirements

A control process consists of two main components as illustrated in Figure 2: a system under control and a controller. In feedback control, the current state of the system is monitored, e.g., using sensors and the controller influences the system based on the difference between the current and the reference state to keep it close to this reference state.

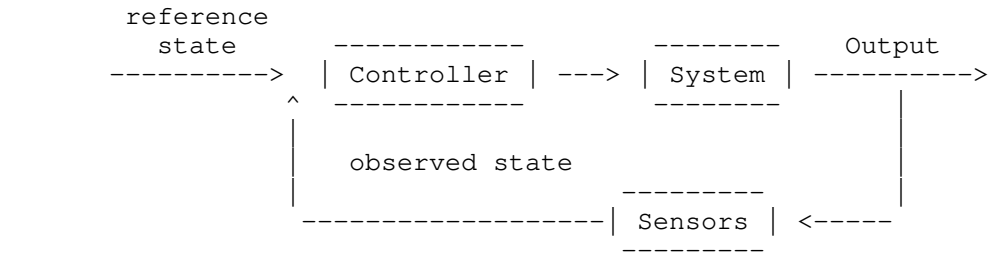

Figure 2: Simple feedback control model.

Apart from the control model, the quality of the control primarily depends on the timely reception of the sensor feedback, because the controller can only react if it is notified of changes in the system state. Depending on the dynamics of the controlled system, the control can be subject to tight latency constraints, often in the single-digit millisecond range. While low latencies are essential, there is an even greater need for stable and deterministic levels of latency, because controllers can generally cope with different levels of latency, if they are designed for them, but they are significantly challenged by dynamically changing or unstable latencies. The unpredictable latency of the Internet exemplifies this problem if off-premise cloud platforms are included.

The main requirements for the industrial control scenario are low and stable latencies to ensure that processes can work continuously and that no machines are damaged.

2.1.1. Approaches

Control models, in general, can become involved but there is a variety of control algorithms that are composed of simple computations such as matrix multiplication. As these are supported by programmable network devices, it is possibile to compose simplified approximations of the more complex algorithms and deploy them in the network. While the simplified versions induce a more inaccurate control, they allow for a quicker response and might be sufficient to operate a basic tight control loop while the overall control can still be exercised from the cloud. The problem, however, is that networking devices typically only allow for integer precision computation while floating-point precision is needed by most control algorithms. Early approaches like [RUETH] have already shown the general applicability of such ideas, but there are still a lot of open research questions not limited to the following:

- How can one derive the simplified versions of the overall controller?

    * How complex can they become?

    * How can one take the limited computational precision of networking devices into account when making them?

  o How does one distribute the simplified versions in the network?

  o How does the overall controller interact with the simplified versions?

3. Large Volume Applications/ Traffic Filtering

In the IIoT, processes and machines can be monitored more effectively resulting in more available information. This data can be used to deploy machine learning (ML) techniques and consequently help to find previously unknown correlations between different components of the production which in turn helps to improve the overall production system. Newly gained knowledge can be shared between different sites of the same company or even between different companies.

Traditional company infrastructure is neither equipped for the management and storage of such large amounts of data nor for the computationally expensive training of ML approaches. Similar to the considerations in Section 2, off-premise cloud platforms offer cost-effective solutions with a high degree of flexibility and scalability. While the unpredictable latency of the Internet is only a subordinate problem for this use case, moving all data to off-premise locations primarily poses infrastructural challenges which are presented in more detail in the following.

3.1. Characterization and Requirements

Processes in the industrial domain are monitored by distributed sensors which range from simple binary (e.g., light barriers) to sophisticated sensors measuring the system with varying degrees of resolution. Sensors can further serve different purposes, as some might be used for time-critical process control while others are only used as redundant fallback platforms. Overall, there is a high level of heterogeneity which makes managing the sensor output a challenging task.

Depending on the deployed sensors and the complexity of the observed system, the resulting overall data volume can easily be in the range of several Gbit/s [GLEBKE]. Using off-premise clouds for managing the data requires uploading or streaming the growing volume of sensor data using the companies' Internet access which is typically limited to a few hundred of Mbit/s. While large networking companies can simply upgrade their infrastructure, most industrial companies rely

on traditional ISPs for their Internet access. Higher access speeds are hence tied to higher costs and, above all, subject to the supply of the ISPs and consequently not always available. A major challenge is thus to devise a methodology that is able to handle such amounts of data over limited access links.

Another aspect is that business data leaving the premise and control of the company further comes with security concerns, as sensitive information or valuable business secrets might be contained in it. Typical security measures such as encrypting the data make in-network computing techniques hardly applicable as they typically work on unencrypted data. Adding security to in-network computing approaches, either by adding functionality for handling encrypted data or devising general security measures, is thus an auspicious field for research which we describe in more detail in Section 5.

3.2. Approaches

There are at least two concepts which might be suitable for reducing the amount of transmitted data in a meaningful way:

1. filtering out redundant or unnecessary data

2. aggregating data by applying pre-processing steps within the network

Both concepts require detailed knowledge about the monitoring infrastructure at the factories and the purpose of the transmitted data.

3.2.1. Traffic Filters

Sensors are often set up redundantly, i.e., part of the collected data might also be redundant. Moreover, they are often hard to configure or not configurable at all which is why their resolution or sampling frequency is often larger than required. Consequently, it is likely that more data is transmitted than is needed or desired. A trivial idea for reducing the amount of data is thus to filter out redundant or undesired data before it leaves the premise using simple traffic filters that are deployed in the on-premise network. There are different approaches to how this topic can be tackled. A first step would be to scale down the available sensor data to the data rate that is needed. For example, if a sensor transmits with a frequency of 5 kHz, but the control entity only needs 1 kHz, only every fifth packet containing sensor data is let through. Alternatively, sensor data could be filtered down to a lower frequency while the sensor value is in an uninteresting range, but let through with higher resolution once the sensor value range

becomes interesting. It is important that end-hosts are informed about the filtering so that they can distinguish between data loss and data filtered out on purpose.

In this context, the following research questions can be of interest:

- How can traffic filters be designed?
- How can traffic filters be coordinated and deployed?
- How can traffic filters be changed dynamically?
- How can traffic filtering be signaled to the end-hosts?

3.2.2. In-Network (Pre-)Processing

There are manifold computations that can be performed on the sensor data in the cloud. Some of them are very complex or need the complete sensor data during the computation, but there are also simpler operations which can be done on subsets of the overall dataset or earlier on the communication path as soon as all data is available. One example is finding the maximum of all sensor values which can either be done iteratively on each intermediate hop or at the first hop, where all data is available.

Using expert knowledge about the exact computation steps and the concrete transmission path of the sensor data, simple computation steps can be deployed in the on-premise network to reduce the overall data volume and potentially speed up the processing time in the cloud.

Related work has already shown that in-network aggregation can help to improve the performance of distributed ML applications [SAPIO]. Investigating the applicability of stream data processing techniques to programmable networking devices is also interesting, because sensor data is usually streamed. In this context, the following research questions can be of interest:

- Which (pre-)processing steps can be deployed in the network?
  - How complex can they become?
- How can applications incorporate the (pre-)processing steps?
- How can the programming of the techniques be streamlined?

4. Industrial Safety (Dead Man's Switch)

Despite increasing automation in production processes, human workers are still often necessary. Consequently, safety measures have a high priority to ensure that no human life is endangered. In traditional factories, the regions of contact between humans and machines are well-defined and interactions are simple. Simple safety measures like emergency switches at the working positions are enough to provide a decent level of safety.

Modern factories are characterized by increasingly dynamic and complex environments with new interaction scenarios between humans and robots. Robots can either directly assist humans or perform tasks autonomously. The intersect between the human working area and the robots grows and it is harder for human workers to fully observe the complete environment.

Additional safety measures are essential to prevent accidents and support humans in observing the environment. The increased availability of sensor data and the detailed monitoring of the factories can help to build additional safety measures if the corresponding data is collected early at the correct position.

4.1. Characterization and Requirements

Industrial safety measures are typically hardware solutions because they have to pass rigorous testing before they are certified and deployment-ready. Standard measures include safety switches, which need to be triggered manually, and light barriers. Additionally, the working area can be explicitly divided into 'contact' and 'safe' areas, indicating when workers have to watch out for interactions with machinery.

These measures are static solutions, potentially relying on specialized hardware, and are challenged by the increased dynamics of modern factories where the factory configuration can be changed on demand. Software solutions offer higher flexibility as they can dynamically respect new information gathered by the sensor systems. Depending on the corresponding occupational safety laws, the software has to satisfy stringent requirements which cannot be satisfied by regular best-effort networks.

4.1.1. Approaches

Software-based solutions can take advantage of the large amount of available sensor data. Different safety indicators within the production hall can be combined within the network so that programmable networking devices can give early responses if a

potential safety breach is detected. A rather simple possibility could be to track the positions of human workers and robots. Whenever a robot gets too close to a human in a non-working area or if a human enters a defined safety zone, robots are stopped to prevent injuries. More advanced concepts could also include image data or combine arbitrary sensor data.

In this context, the following research questions can be of interest:

- How can the software give guaranteed safety over best-effort networks?
- Which sensor information can be combined and how?

5. Security Considerations

Current in-network computing approaches typically work on unencrypted plain text data because today's networking devices usually do not have crypto capabilities. As is already mentioned in Section 3.1, this above all poses problems when business data, potentially containing business secrets, is streamed into remote computing facilities and consequently leaves the control of the company. Insecure on-premise communication within the company and on the shop-floor is also a problem as machines could be intruded from the outside. It is thus crucial to deploy security and authentication functionality on on-premise and outgoing communication although this might interfere with in-network computing approaches. Ways to implement and combine security measures with in-network computing are described in more detail in [I-D.fink-coin-sec-priv].

6. IANA Considerations

N/A

7. Conclusion

In-network computing concepts have the potential to improve industrial applications. There are at least three scenarios for which in-network processing can be beneficial, each having a unique set of requirements.

In the control scenario, tight latency constraints in the single digit millisecond range have to be satisfied despite the use of cloud platforms and the corresponding unstable latency of the Internet.

In a second scenario, large amounts of data have to be transmitted to cloud platforms for further evaluation. One important task here is to reduce the amount of data that needs to be transmitted as the

available Internet access speed is most likely non-sufficient. Apart from that, security measures have to be implemented as business data is transmitted to the Internet.

Regarding safety, software-based measures often lack the required guarantees and do not withstand the testing for certification. In-network processing with its potential for early responses can be a solution by combining different sensor outputs early and acting quickly.

8. Informative References

[GLEBKE] Glebke, R., "A Case for Integrated Data Processing in Large-Scale Cyber-Physical Systems", DOI: 10125/60162, in HICSS, January 2019.

[I-D.fink-coin-sec-priv] Fink, I. and K. Wehrle, "Enhancing Security and Privacy with In-Network Computing", draft-fink-coin-sec-priv-00 (work in progress), March 2020.

[I-D.mcbride-edge-data-discovery-overview] McBride, M., Kutscher, D., Schooler, E., and C. Bernardos, "Edge Data Discovery for COIN", draft-mcbride-edge-data-discovery-overview-03 (work in progress), January 2020.

[RUETH] Rueth, J., "Towards In-Network Industrial Feedback Control", DOI: 10.1145/3229591.3229592, in ACM SIGCOMM NetCompute, August 2018.

[SAPIO] Sapio, A., "Scaling Distributed Machine Learning with In-Network Aggregation", 2019, <https://arxiv.org/abs/1903.06701>.

[TSN] "Time-Sensitive Networking (TSN) Task Group", 2019, <https://1.ieee802.org/tsn/>.

Authors' Addresses

Ike Kunze
RWTH Aachen University
Ahornstr. 55
Aachen D-50274
Germany

Phone: +49-241-80-21422
Email: kunze@comsys.rwth-aachen.de

Klaus Wehrle
RWTH Aachen University
Ahornstr. 55
Aachen D-50274
Germany

Phone: +49-241-80-21401
Email: wehrle@comsys.rwth-aachen.de

COINRG
Internet-Draft
Intended status: Informational
Expires: September 10, 2020

I. Kunze
K. Wehrle
RWTH Aachen University

March 09, 2020

Transport Protocol Issues of In-Network Computing Systems
draft-kunze-coinrg-transport-issues-01

Abstract

Today's transport protocols offer a variety of functionality based on the notion that the network is to be treated as an unreliable communication medium. Some, like TCP, establish a reliable connection on top of the unreliable network while others, like UDP, simply transmit datagrams without a connection and without guarantees into the network. These fundamental differences in functionality have a significant impact on how COIN approaches can be designed and implemented. Furthermore, traditional transport protocols are not designed for the multi-party communication principles that underlie many COIN approaches. This document discusses selected characteristics of transport protocols which have to be adapted to support COIN functionality.

Table of Contents

1. Introduction

A fundamental design choice of the Internet is that the network should be kept as simple as possible while complexity in the form of processing should be located on end-hosts at the edges of the network. This choice is reflected in the end-to-end principle which states that end-hosts directly address each other and perform all relevant computations while the network only delivers the packets without modifying them. Transport protocols are consequently designed to facilitate the direct communication between end-hosts.

In practice, the end-to-end principle is often violated by intransparent middleboxes which alter transmitted packets, e.g., by dropping or changing header fields. Contrary to that, computing in the network (COIN) encourages explicit computations in the network which introduces an intertwined complexity as the computations on the end-hosts depend on the functionality deployed in the network. It further challenges traditional end-to-end transport protocols as they are not designed to address in-network computation entities or to include more than two devices into a communication. Some of these problems are already presented in [I-D.draft-kutscher-coinrg-dir-01].

This draft intends to discuss potential problems for traditional transport protocols in more detail to raise questions that the COIN

community needs to solve before a widespread application of COIN functionality is sensible. Collaboration with other IRTF and IETF groups, such as PANRG, the IETF transport area in general, or the LOOPS BOF, can help in finding suitable solutions.

2. Addressing

The traditional addressing concept of the Internet is that end-hosts directly address each other with all computational intelligence residing at the network edges. With COIN, computations move into the network and need to be integrated into the established infrastructure. In systems where the whole network is under the control of the network operator this integration can be implemented by explicitly adjusting the communication schemes based on the COIN functionality. Considering larger scales, e.g., Internet-wide deployment, this approach of manually adjusting traffic patterns and applications to correctly incorporate changes made by the network is no longer feasible.

What is needed are ways to specify which kind of functionality should be applied to the transmitted data on the path inside the network and maybe even where or by whom the execution should take place. Such functionality could for example be implemented using an indirection mechanism which routes a packet along a pre-defined or dynamically chosen path which then realizes the desired functionality. Related concepts are Segment and Source Routing as well as (Service/Network) Function Chaining/Composition.

Main challenges/questions are:

1. How should end-hosts address the COIN functionality?

2. How can the end-hosts influence where or by whom the functionality is executed?

3. How can devices which do not implement COIN functionality be integrated into the systems without breaking the COIN or legacy functionality?

Another question is how the transmitted data is to be treated by the devices implementing the functionality.

3. Flow granularity

Core networking hardware pipelines such as backbone switches are built to process incoming packets on a per-packet basis, keeping little to no state between them. This is appropriate for the general task of forwarding packets, but might not be sufficient for COIN as

information that is needed for the computations can be spread across several packets. In a TCP stream, for example, data is dynamically distributed across different segments which means that the data needed for application-level computations might also be split up. In contrast to that the content of UDP datagrams is defined by the application itself which is why the datagrams could either be self-contained or information can be cleverly distributed onto different datagrams. The common scheme is that different transport protocols induce different meanings to the packets that they send out which needs to be accounted for in COIN elements as they have to know how the received data is to be interpreted. There are at least three options for this.

1. Every packet is treated individually. This respects the possibilities that are already offered by all networking equipment.

2. Every packet is treated as part of a message. In this setting, the packet alone does not have enough information for the computations. Instead, it is important to know the content of the surrounding packets which together form the overall message.

3. Every packet is treated as part of a byte stream. Here, all previous packets and potentially even all subsequent packets need to be taken into consideration for the computations as the current packet could, e.g., be the first of a group of packets, a packet in the middle or the final packet.

The flow granularity consequently has a significant impact on how computations can be performed and where. Apart from how the COIN elements should treat the transmitted data, another important aspect is how it can be ensured that the end-hosts know who has altered the data and how.

4. Authentication

The realisation of COIN legitimizes and actively promotes that data transmitted from one host to another can be altered on the way inside the network. This opens the door for foul play as all intermediate network elements - no matter if they are malicious or misbehaving by accident, COIN elements, or 'traditional' middleboxes - could simply start altering parts of the original data and potentially cause harm to the end-hosts. What is needed are mechanisms with which the receiving host can verify (a) how and (b) by whom the data has been altered on the way. In fact, these might very well be two distinct mechanisms as one (a) only focusses on the changes that are made to the data while (b) requires a scheme with which COIN elements can be uniquely identified (could very well relate to Section 2) and

subsequently authenticated. The Proof of Transit [I-D.draft-ietf-sfc-proof-of-transit-04] concept of the SFC WG could be applicable for proving that a packet has indeed passed the desired COIN elements, although it does not provide means to validate which changes were made by the known nodes.

Main challenges/questions are:

1. How are changes to the data within the network communicated to the end-hosts?

2. How are the COIN elements that are responsible for the changes communicated to the end-hosts?

3. How are changes made by the COIN elements authenticated?

5. Security

Many early COIN concepts require an unencrypted transmission of data. At the same time, there is a general tendency towards more and more security features in communication protocols. QUIC, e.g., encrypts all payload data and almost all header content already inside the transport layer. This makes current COIN concepts infeasible in settings where QUIC connections are used as the COIN elements do not have access to any packet content. Using COIN thus also depends on how well security mechanisms like encryption can be integrated into COIN frameworks.

Together, the four aspects presented in Section 2 to Section 5 form a set of fundamental properties for a basic transport-compatible realization of COIN. In the following, we briefly discuss selected additional transport features to create awareness for the multifaceted interaction between the transport protocols and COIN elements.

6. Advanced Transport Features

Depending on application needs, different transport protocols provide different features. These features shape the behavior of the protocol and have to be taken into account when developing COIN functionality. In this section, we focus on the impact of reliability as well as flow and congestion control.

6.1. Reliability

Applications require a reliable transport whenever it is important that all data has been transmitted successfully. TCP[RFC0793] provides such a reliable communication as it first sets up a

dedicated connection and then ensures the successful reception of all data. In contrast, UDP[RFC0768] is a connectionless protocol without guarantees and COIN elements working on UDP transmissions must be robust to lost information. This is not the case for applications on top of TCP, but the retransmissions and the TCP state, which TCP uses to achieve the reliability, make packet processing for COIN more complex due to at least three reasons.

The concept of retransmissions bases on the end-to-end principle as retransmissions are performed by the sender if it has determined that the receiver did not receive the corresponding original message. Both participants can then act knowing that parts of the overall data are still missing. For simple COIN elements, which are not aware of the involved TCP states and which do not track sequence numbers, it is difficult to identify (a) that a packet in the sequence is missing and (b) that a packet is a retransmission. One question is whether COIN elements should incorporate an understanding for retransmissions on the basis of existing transport mechanisms or if a COIN-capable transport should include dedicated signals for the COIN elements.

Apart from challenges in identifying retransmissions, there is also the fact that they are sent out of order with the original packet sequence. Depending on the chosen flow granularity (see Section 3), COIN elements might have to hold contextual information for a prolonged time once they identify an impeding retransmission. Moreover, they might have to postpone or cancel computations if data is missing and instead schedule later computations. The main question arising from this is: to what extent should COIN elements be capable of incorporating retransmissions into their computation schemes and how much additional storage capabilities are required for this?

When incorporating COIN elements into the retransmission mechanisms, it is also an interesting question whether it should be possible to request or perform retransmissions from COIN elements. Considering a setting with COIN elements that are capable of detecting missing packets and retransmission requests, it might improve the overall performance if the COIN element directly requests or performs the retransmission instead of forwarding the packet/request through the complete sequence of elements. In both cases, the aforementioned storage capabilities are relevant so that the COIN elements can store enough information. The general question, i.e., which nodes in the sequence should do the retransmission, has already been worked on in the context of multicast transport protocols.

Depending on the extent of realization of the presented retransmission features, COIN elements might almost have to implement some of TCP's state to fulfil their tasks. Considering that

different COIN elements have different computational and storage capacities, it is very likely that not every form of transport integration into COIN can be supported by every available COIN platform. The choice of devices included into the communication will hence certainly affect the types of transport protocols that can be operated on the COIN networks.

Challenges/questions regarding reliability are:

1. Should COIN elements be aware of retransmissions?

2. How can COIN elements identify missing packets or retransmissions?

3. Should COIN elements be explicitly notified about retransmissions?

4. To what extent should COIN elements be capable of incorporating retransmissions into their computation schemes?

5. How much storage capabilities are required for incorporating retransmissions?

6. How can COIN elements incorporate missing packets into their computations?

7. How to deal with state changes in COIN elements caused by data lost later in the communication chain and then retransmitted?

8. Should COIN elements be capable of requesting retransmissions/ answering retransmission requests?

9. Which devices should perform retransmissions?

10. Do COIN elements have to keep transport state?

11. How much transport state do COIN elements have to keep?

Another aspect is flow and congestion control to avoid overloading the receiving end-host and the network.

6.2. Flow/Congestion Control

TCP incorporates mechanisms to avoid overloading the receiving host (flow control) and the network (congestion control) and determines its sending rate as the minimum value of what both mechanisms determine as feasible for the system. This approach is based on the notion that computing and forwarding hosts are separated and is

challenged by the inclusion of COIN elements, i.e., computing nodes in the network.

Flow control bases on explicit end-host information as the participating end-hosts notify each other about how much data they are capable of processing and consequently do not transmit more data as the other host can handle. This only changes if one of the end-hosts updates its flow control information.

Congestion control, on the other hand, interprets volatile feedback from the network to guess a sending rate that is possible given the current network conditions. Most congestion control algorithms hereby follow a cyclical procedure where the sending end-hosts constantly increase their sending rate until they detect network congestion. They then decrease their sending rate once and start to increase it again.

In this traditional two-fold approach, loss, delay, or any other congestion signal (depending on the congestion control algorithm) induced by COIN elements (only in case that they are the bottleneck) is interpreted as network congestion and thus accounted for in the congestion control mechanism. This means that the sending end-host may repeatedly overload the computational capabilities of the COIN elements when probing for the current network conditions instead of respecting general device capabilities as is done by flow control.

Consequently, the question arises whether COIN elements should be able to participate in end-to-end flow control.

Main challenges/questions are:

1. Should COIN elements be covered by congestion control?

2. Should COIN elements be able to participate in end-to-end flow control?

3. How could a resource constraint scheme similar to flow control be realized for COIN elements?

All in all, there is a wide range of non-essential transport features which offer improved performance in certain settings and for certain application combinations. However, as presented, it is likely that not all of the features/types of transport protocols can be supported on every COIN element. It might make sense to define different classes of COIN-ready transport protocols which can then be deployed depending on the concretely available networking/hardware elements. Alternatively, each of the features could be treated separately and

they could then be composed based on the demands of an application at-hand.

Overall, the general question summarizing all of the other challenges/question is:

Which transport features should be supported by COIN, how can they be identified and how can they be provided to application designers?

7. Security Considerations

TBD

8. IANA Considerations

N/A

9. Conclusion

The advent of COIN comes with many new use cases and promises improved solutions for various problems. It is, however, not directly compatible with the end-to-end nature of transport protocols. To enable a transport-based communication, it is thus important to answer key questions regarding COIN and transport protocols, some of which are raised in this document.

10. Informative References

[I-D.draft-ietf-sfc-proof-of-transit-04]
Brockners, F., Bhandari, S., Mizrahi, T., Dara, S., and S. Youell, "Proof of Transit", draft-ietf-sfc-proof-of-transit-04 (work in progress), November 2019.

[I-D.draft-kutscher-coinrg-dir-01]
Kutscher, D., Karkkainen, T., and J. Ott, "Directions for Computing in the Network", draft-kutscher-coinrg-dir-01 (work in progress), November 2019.

[RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, DOI 10.17487/RFC0768, August 1980, <https://www.rfc-editor.org/info/rfc768>.

[RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, DOI 10.17487/RFC0793, September 1981, <https://www.rfc-editor.org/info/rfc793>.

Authors' Addresses

Ike Kunze
RWTH Aachen University
Ahornstr. 55
Aachen D-50274
Germany

Phone: +49-241-80-21422
Email: kunze@comsys.rwth-aachen.de

Klaus Wehrle
RWTH Aachen University
Ahornstr. 55
Aachen D-50274
Germany

Phone: +49-241-80-21401
Email: wehrle@comsys.rwth-aachen.de

COINRG D. Kutscher
Internet-Draft University of Applied Sciences Emden/Leer
Intended status: Experimental T. Kaerkkaeinen
Expires: February 1, 2021 J. Ott
Technical University Muenchen
July 31, 2020

Directions for Computing in the Network
draft-kutscher-coinrg-dir-02

Abstract

In-network computing can be conceived in many different ways - from active networking, data plane programmability, running virtualized functions, service chaining, to distributed computing.

This memo proposes a particular direction for Computing in the Networking (COIN) research and lists suggested research challenges.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on February 1, 2021.

Copyright Notice

Table of Contents

1. Introduction

Recent advances in platform virtualization, link layer technologies and data plane programmability have led to a growing set of use cases where computation near users or data consuming applications is needed - for example, for addressing minimal latency requirements for compute-intensive interactive applications (networked Augmented Reality, AR), for addressing privacy sensitivity (avoiding raw data copies outside a perimeter by processing data locally), and for speeding up distributed computation by putting computation at convenient places in a network topology.

In-network computing has mainly been perceived in five variants so far: 1) Active Networking [ACTIVE], adapting the per-hop-behavior of network elements with respect to packets in flows, 2) Edge Computing as an extension of virtual-machine (VM) based platform-as-a-service, 3) programming the data plane of SDN switches (through powerful programmable CPUs and programming abstractions, such as P4 [SAPIO]), 4) application-layer data processing frameworks, and 5) Service Function Chaining (SFC).

Active Networking has not found much deployment in the past due to its problematic security properties and complexity.

Programmable data planes can be used in data centers with uniform infrastructure, good control over the infrastructure, and the feasibility of centralized control over function placement and scheduling. Due to the still limited, packet-based programmability model, most applications today are point solutions that can demonstrate benefits for particular optimizations, however, often without addressing transport protocol services or data security that would be required for most applications running in shared infrastructure today.

Edge Computing (in the ETSI Multi-access Edge Computing [MEC] variant, as traditional cloud computing) has a fairly coarse-grained (VM-based) computation-model and is hence typically deploying centralized positioning/scheduling though virtual infrastructure management (VIM) systems. Besides such industry-driven activities, manifold research approaches to edge computing with varying granularity and orchestration approaches, among other differiating elements, have been pursued [EDGESURVEY] [FOGEDGE].

Microservices can be seen as a (lightweight) extension of the cloud computing model (application logic in containers and orchestrators for resource allocation and other management functions), leveraging more lightweight platforms and fine-grained functions. Compared to traditional VM-based systems, microservice platforms typically employ a "stateless" approach, where the service/application state is not tied to the compute platform, thus achieving fault tolerance with respect to compute platform/process failures.

Application-layer data processing such as Apache Flink [FLINK] provide attractive dataflow programming models for event-based stream processing and light-weight fault-tolerance mechanisms - however systems such as Flink are not designed for dynamic scheduling of compute functions.

Modern distributed applications frameworks such as Ray [RAY], Sparrow [SPARROW] or Canary [CANARY] are more flexible in this regard - but

since they are conceived as application-layer frameworks, their scheduling logic can only operate with coarse-granular cost information. For example, application-layer frameworks in general, can only infer network performance, anomalies, optimization potential indirectly (through observed performance or failure), so most scheduling decisions are based on metrics such as platform load.

Service Function Chaining (SFC, [RFC7665]) is about establishing IP tunnels between processing functions that are expected to work on packets or flows - for applications such as inspection and classification, so that some of these functions could be seen as elements in a COIN context as well.

2. Terminology

We are using the following terms in this memo:

Program: a set of computations requested by a user

Program Instance: one currently executing instance of a program

Function: a specific computation that can be invoked as part of a program

Execution Platform: a specific host platform that can run function code

Execution Environment: a class of target environments (execution platforms) for function execution, for example, a JVM-based execution environment that can run functions represented in JVM byte code

3. Computing in the Network vs Networked Computing vs Packet Processing

Many applications that might intuitively be characterized as "computing in the network" are actually either about connecting compute nodes/processes or about IP packet processing in fairly traditional ways.

Here, we try to contrast these existing and widely successful systems (that probably do not require new research) with a more novel "computing in the network (COIN)" approach that revisits the function split between computing and networking.

3.1. Networked Computing

Networked Computing exists in various facets today (as described in the Introduction). Fundamentally, these systems make use of networking to connect compute instances - be it VMs, containers, processes or other forms of distributed computing instances.

There are established frameworks for connecting these instances, from general purpose Remote Method Invocation/Remote Procedure Calls to system-specific application-layer protocols. With that, these systems are not actually realizing "computing in the network" - they are just using the network (and taking connectivity as granted).

Most of the challenges here are related to compute resource allocation, i.e., orchestration methods for instantiating the right compute instance on a corresponding platform - for achieving fault tolerance, performance optimization and cost reduction.

Examples of successful applications of networked computing are typical overlay systems such as CDNs. As overlays they do not need to be "in the network" - they are effectively applications. (Note: we sometimes refer to CDN as an "in-network" service because of the mental model of HTTP requests that are being directed and potentially forwarded by CDN systems. However, none of this happens "in the network" - it is just a successful application of HTTP and underlying transport protocols.)

3.2. Packet Processing

Packet processing is a function "in the network" - in a sense that middleboxes reside in the network as transparent functions that apply processing functions (inspection, classification, filtering, load management etc.) - mostly _transparent_ to endpoints. Some middlebox functions (TCP split proxies, video optimizers) are more invasive in a sense that they do not only operate on IP flows but also try to impersonate transport endpoints (or interfere with their behavior).

Since these systems can have severe impacts on service availability, security/privacy, and performance they are typically not very _programmable_ - they just execute (usually) static code for predefined functions.

Active Networking can be characterized as an attempt to offer abstractions for programmable packet processing from an "endpoint perspective", i.e., by using data packets to specify intended behavior in the network with the aforementioned security problems.

Programmable Data Plane approaches such as P4 are providing abstractions of different types of network switch hardware (NPUs, CPUs, FPGA, PISA) from a switch/network programming perspective. The corresponding programs are constrained by the capabilities (instruction set, memory) of the target platform and typically operate on packets/flow abstractions (for example match-action-style processing).

Network Functions Virtualization (NFV) is essentially a "Networked Computing" approach (after all, Network Functions are just virtualized compute functions that get instantiated on compute platforms by an orchestrator). However, some Virtual Network Functions (VNFs) happen to process/forward packets (e.g., gateways in provider networks, NATs or firewalls). Still, that does not affect their fundamental properties as virtualized computing functions.

When connecting VNFs, there is the question of how to steer packet flows so that packets reach the right functions (and pass through them in the right order). One way is through configuration and network control/management (SDN), i.e., the VNFs are places in a virtual network, and there are configurations for meaningful next-hop IP addresses etc.

A more dynamic way is through Service Function Chaining (SFC, [RFC7665]), where a dynamic chain of IP-addressable packet processors can be specified (in an encapsulation packet header structure) and where forwarding nodes are equipped to interpret these headers and forward the packets to the appropriate next hops.

The SFC [RFC7665]) framework works with IP addresses for function (host) identifiers. Name-Based Service Function Forwarding [I-D.trossen-sfc-name-based-sff] takes this one step further by adding another layer of indirection and by identifying the Service Functions using a name rather than a routable IP endpoint (or Layer 2 address). In addition to the naming concept, [I-D.trossen-sfc-name-based-sff] also described the possibility of using different transport and application layer protocols for the communication between functions - which could in principle extend the applicability from mere packet processing to some form of distributed computing.

3.3. Computing in the Network

In some deployments, networked computing and packet processing go well together, for example, when network virtualization (multiplexing physical infrastructure for multiple isolated subnetworks) is achieved through data-plane programming (SDN-style) to provide connectivity for VMs of a tenant system.

While such deployments are including both computing and networking, they are not really doing computing _in the network_. VM/containers are virtualized hosts/processes using the existing network, and packet processing/programmable networks is about packet-level manipulation. While it is possible to implement certain optimizations (for example, processing logic for data aggregation) - the applicability is rather limited especially for applications where application-data units do not map to packets and where additional transport protocols and security requirements have to be considered.

Multi-access Edge Computing [MEC] is a particular architecture that leverages the virtual host platform concept, and that is focused on management and orchestration for such platforms. MEC can be combined with virtual networking concepts such as "Network Slicing" in 5G [MEC5G] to assure a certain QoS for connectivity to MEC platform instances. It should be noted that there may be other forms of edge computing that are not VM-based.

Distributed Computing (stream processing, edge computing) on the other side is an area where many application-layer frameworks exist that actually _could_ benefit from a better integration of computing and networking, i.e., from a new "computing in the network" approach.

For example, when running a distributed application that requires dynamic function/process instantiation, traditional frameworks typically deploy an orchestrator that keeps track of available host platforms and assigned functions/processes. The orchestrator typically has good visibility of the availability of and current load on host platforms, so it can pick suitable candidates for instantiating a new function.

However, it is typically agnostic of the network itself - as application layer overlays the function instances and orchestrators take the network as a given, assuming full connectivity between all hosts and functions. While some optimizations may still be feasible (for example co-locating interacting functions/processes on a single host platform), these systems cannot easily reason about

- shortest paths between function instances; function off-loading
- opportunities on topologically convenient next hops; and
- availability of new, not yet utilized resources in the network.

While it is possible to perform optimizations like these in application layers overlays, it involves significant monitoring effort and would often duplicate information (topology, latency) that is readily available inside the network. In addition to the

associated overhead, such systems also operate at different time scales so that direct reaction in fine-grained computing environments is difficult to achieve.

When asking the question of how the network can support distributed computing better, it may be helpful to characterize this problem as a resource allocation optimization problem: Can we integrate computing and networking in a way that enables a joint optimization of computing and networking resource usage? Can we apply this approach to achieve certain optimization goals such as:

- low latency for certain function calls or compute threads;
- high throughput for a pipeline of data processing functions;
- high availability for an overall application/service;
- load management (balancing, concentration) according to performance/cost constraints; and
- consideration of security/privacy constraints with respect to platform selection and function execution?
- Also: can we do this at the speed of network dynamics, which may be substantially higher than the rate at which distributed computing applications change?

Considering computing and networking resource holistically could be the key for achieving these optimization goals (without considerable overhead through telemetry, management and orchestration systems). If we are able to dissolve the layer boundaries between the networking domain (that is typically concerned with routing, forwarding, packet/flow-level load balancing) and the distributed computing domain (that is typically concerned with 'processor' allocation, scaling, reaction to failure for functions/processes), we might get a handle to achieve a joint resource optimization and enable the distributed computing layer to leverage network-provided mechanisms directly.

For example, if distributing information about available/suitable compute platform could be a routing function, we might be able to obtain and utilize this information in a distributed fashion. If instantiating a new function (or offloading some piece of computation) could consider live performance data obtained from a in-network forwarding/offloading service (similar to IP packet forwarding in traditional IP networks), the "next-hop" decision could be based both on network performance and node load/availability).

Integrating computing and networking in this manner would not rule out highly optimized systems leveraging sophisticated orchestrators. Instead, it would provide a (possibly somewhat uniform) framework that could allow several operating and optimization modes, including totally distributed modes, centralized orchestration, or hybrid forms, where policies or intents are injected into the distributed decision-making layer, i.e., as parameters for resource allocation and forwarding decisions.

3.4. Elements for Computing in the Network

In-network computing requires computing resources (CPU, possibly GPUs, memory, ...), physical or virtualized to some extent by a suitable platform. These computing resources may be available in a number of places, as partly already discussed above, including the following:

- They may be found on dedicated machines co-locating with the routing infrastructure, e.g., having a set of servers next to each router as one may find in access network concentrators. This would come closest to today's principles of edge computing.

- They may be integrated with routers or other network operations infrastructure and thus be tightly integrated within the same physical device.

- They may be integrated within switches, similar to the (limited) P4 compute capabilities offered today.

- They may be located on NICs (in hosts) or line cards (routers) and be able to proactively perform some application functions, in the sense of a generalized variant of "offloading" that protocol stacks perform to reduce main CPU load.

- They might add novel types of dedicated hardware to execute certain functions more efficiently, e.g., GPU nodes for (distributed) analytics.

- They might include low-end (embedded) devices such as microcontrollers that support decentralized computation at low cost and limited performance.

- They may also encompass additional resources at the edge of the network, such as sensor nodes. Associated sensors could be physical (as in IoT) or logical (as in MIB data about a network device).

- Even user devices along the lines of crowd computing [CROWD] or mist computing [MIST] may contribute compute resources and dynamically become part of the network.

Depending on the type of execution platform, as already alluded to above, a suitable execution framework must be put in place: from lambda functions to threads to processes or process VMs to unikernels to containers to full-blown VMs. This should support mutual isolation and, depending on the service in question, a set of security features (e.g., authentication, trustworthy execution, accountability). Further, it may be desirable to be able to compose the executable units, e.g., by chaining lambda functions or allowing unikernels to provide services to each other - both within a local execution platform and between remote platform instances across the network.

The code to be executed may be pre-installed (as firmware, as microcode, as operating system functions, as libraries, as *aaS offering, among others) or may be dynamically supplied. While the former is governed by the entity operating the execution device or supplying it (the vendor), the code to be executed may have different origins. Fundamentally, we can distinguish between two cases:

1. The code may be "centrally" provisioned, originating from an application or other service provider inside the network. This is analogous to CDNs, in which an application provider contracts a CDN provider to host content and service logic on its behalf. The deployment is usually long-term, even if instantiations of the code may vary. The code thus originates from rather few - known - sources. In this setting, applications only invoke this code and pass on their parameters, context, data, etc.

2. The code may be provided in a decentralized manner from a user device or other service that requires a certain function or service to be carried out. At the coarse granularity of entire application images, this has been explored as "code offloading"; recent approaches have moved towards finer granularities of offloading (sets of) functions, for which also some frameworks for smartphones were developed, leading to finer granularities down to individual functions. In this setting, application transfer mobile code - along with suitable parameters, etc. - into the network that is executed by suitable execution platforms. This code is naturally expected to be less trusted as it may come from an arbitrary source.

Obviously, 1. and 2. may be combined as mobile code may make use of other in-network functions and services, allowing for flexible application decomposition. Essentially, computing in the network may

support everything from full application offloading to decomposing an application into small snippets of code (e.g., at class, objects, or function granularity) that are fully distributed inside the network and executed in a distributed fashion according to the control flow of the application. This may lead to iterative or recursive calling from application code on the initiating host to mobile code to pre-provisioned code.

Another dimension beyond where the code comes from is how tightly the code and the data are coupled. At one extreme, approaches like Active Messages combine the data and the code that operates (only) on that data into transmission units, while at the other extreme approaches like Network Function Virtualization are only concerned with the instantiation of the code in the network. The underlying architectural question is whether the goal is to enable the network to perform computations on the data passing through it, or whether the goal is to enable distributed computational processes to be built in the network. And, of course, complete applications may leverage both approaches.

With these different existing and possibly emerging platforms and execution environments and different ways to provision functions in the network, it does not seem useful to assume any particular platform and any particular "mobile code" representation as _the_ "computing in the network" environment. Instead, it seems more promising to reason about properties that are relevant with respect to distributed program semantics and protocols/interfaces that would be used to integrate functions on heterogeneous platforms into one application context. We discuss these ideas and associated challenges in the following section.

4. Examples

4.1. Compute-First Networking with ICN

[CFN] is an example of a computing-in-the-network system that is based on computation graph representation for distributed programs. These programs are composed of stateful actors and stateful functions that are dynamically instantiated on available compute resources.

The first motivating use case was a real-time health monitoring system that analyzed audio samples from coughing noises which involves processing several audio feeds for noise addition and subtraction and for feature extraction.

The key concept of CFN is to provide a general-purpose distributed computing framework that can be programmed without knowledge about

the runtime environment but that can leverage the dynamic resource properties automatically, and with reasonable efficiency.

CFN can lay out compute graphs over the available computing platforms in a network to perform flexible load management and performance optimizations, taking into account function/actor location and data location, as well as platform load and network performance.

In CFN, compute nodes that can execute functions within a given program instance are called workers. The allocation of functions and actors to workers happens in a distributed fashion. A CFN system knows the current utilization of available resources and the least cost paths to copies of needed input data. It can dynamically decide which worker to use, performing optimizations such as instantiating functions close to big data inputs. The bindings that control which execution platforms host which program interfaces (or individual functions/actors) is maintained through a computation graph.

To realize this distributed scheduling, workers in each resource pool advertise their available resources. This information is shared among all workers in the pool. A worker execution environment can decide, without a centralized scheduler, which set of workers to prefer to invoke a function or to instantiate an actor. In order to direct function invocation requests to specific worker nodes, CFN utilizes the underlying ICN network's forwarding capabilities - the network performs late binding through name-based forwarding and workers can provide forwarding hints to steer the flow of work.

4.2. Akka Toolkit

The Akka toolkit [1] for building concurrent and distributed applications on the the JVM that is used by frameworks such as Apache Flink [2]. Akka is implements the Actor model, a way of realizing distributed computing as asynchronous message-based communication between concurrent processes that encapsulate application logic.

Communication between distributed actors is based on symmetric peer-to-peer model (actors can send each other messages) and is implemented by TCP-based protocols [3].

Akka actors are logically organized in a tree hierarchy [4], and there are two addressing concepts: 1) Actor References that unique identify an actor instance and 2) Actor Paths, hierarchically structured names that specify the logical position of an actor instance in system tree. Actor path can have an address component that specified location information (e.g., host and port number).

Akka has a routing concept [5] that can duplicate and distribute messages to a set of actors (for example for map-reduce like parallelism).

The Akka toolkit support cluster features [6], i.e., the management of a collection of JVMs that can be monitored for resource and failure management.

5. Research Challenges

Conceiving computing in the network as a joint resource optimization problem as described above incurs a set of interesting, novel research challenges that are particularly relevant from an Internet Research perspective.

5.1. Categorization of Different Use Cases for Computing in the Network

There are different applications but also different configuration classes of Computing in the Network systems. For example, a data processing pipeline might be different from a distributed application employing some stateful actor components. It is worthwhile analyzing different typical use cases and identify commonalities (for example, fundamental protocol elements etc.) and differences.

5.2. Networking and Remote-Method-Invocation Abstractions

In distributed systems, there are different classes of functions that can be distinguished, for example:

1. Strictly stateless functions that do not keep any context state beyond their activation time

2. Stateful functions/modules/programs that can be instantiated, invoked and eventually destroyed that do keep state over a series of function invocations

Modern frameworks such as Ray are offering a clear separation of stateless functions and stateful actors and offer corresponding abstractions in their programming environment. The aforementioned analysis of use cases should provide a diverse set of use cases for deriving a minimal yet sufficient set of function classes.

Beyond this fundamental categorization of functions/actors, there is the question of interfaces and protocols mechanisms - as building blocks to utilize functions in programs. For example, stateful functions are typically invoked through some Remote Method Invocation (RMI) protocol that identifies functions, allows for specifying/ transferring parameters and function results etc. Stateful actors

could provide class-like interfaces that offer a set of functions (some of which might manipulate actor state).

Another aspect is about identity (and naming) of functions and actors. For actors that are typically used to achieve real-world effects or to enable multiple invocations of functions manipulating actor state over time, it is obvious that there needs to be a concept of specific instances. Invoking an actor function would then require specifying some actor instance identifier.

Stateless functions may be different: an invoking instance may be oblivious with respect to the specific function instance and locus (on an execution platform) and might just want to leave it to the network to find the "best" instance or locus for a new instantiation. Some fine-granular functions might just be instantiated for one invocation. On the other hand, a function might be tied to a particular execution platform, for example an GPU-supported host system. The naming and identity framework must allow for specifying such a function (or at least equivalence classes) accordingly.

Stateful functions may share state within the same program context, i.e., across multiple invocations by the same application (as, e.g., holds for web services that preserve context - locally or on the client side). But stateful functions may also hold state across applications and possibly across different instantiations of a function on different compute nodes. Such will require data synchronization mechanisms and the implementation of suitable data structure to achieve a certain degree of consistency. The targeted degree of consistency may vary depending on the function and so may the mechanisms used to achieve the desired consistency.

Finally, execution platforms will require efficient resource management techniques to operate with different types of stateless and stateful functions and their associated resources, as well as for dynamically instantiated mobile code. Besides the aforementioned location of suitable compute platforms and scheduling (possibly queuing) functions and function invocations, this also includes resource recovery ("garbage collection").

5.3. Transport Abstractions

When implementing Computing in the Network and building blocks such as function invocation it seems that IP packet processing is not the right abstraction. First of all, carrying the context for some function invocation might require many IP packets - possibly something like Application Data Units (ADUs). But even if such ADUs could be fit into network layer packets, other problems still need to

be addressed, for example message formats, reliability mechanisms, flow and congestion control etc.

It could be argued that today's distributed computing overlays solve that by using TCP and corresponding application layer formats (such as HTTP) - however this begs the question whether a fine-granular distributed computing system, aiming to leverage the network for certain tasks, is best served by a TCP/IP-based approach that entails issues such as

- need for additional resolution/mapping system to find IP addresses for functions;
- possible overhead for establishing TCP connections for fine-granular function invocation;
- defining and managing security properties of such connections and coping with the associated setup/validation overhead; and
- mismatch between TCP end-to-end semantics and the intention to defer next-hop selection etc. to the network.

Moreover, some Computing in the Network applications such as Big Data processing (Hadoop-style etc.) can benefit significantly from data-oriented concepts such as

- in-network caching (of data objects that represent function parameters or results);
- reasoning about the tradeoffs between moving data to function vs. moving code to data assets; and
- sharing data (e.g., function results) between sets of consuming entities.

RMI systems such as RICE [RICE] enable Remote Method Invocation of ICN (data-oriented network/transport). Research questions include investigating how such approaches can be used to design general-purpose distributed computing systems. More specifically, this would involve questions such as:

- What is the role of network elements in forwarding RMI requests?
- What visibility into load, performance and other properties should endpoints and the network have to make forwarding/offloading decisions and how can such visibility be afforded?

- What is the notion of transport services in this concept and how intertwined is traditional transport with RMI invocation?

- What kind of feedback mechanisms would be desirable for supporting corresponding transport services?

5.4. Programming Abstractions

When creating SDKs and programming environments (as opposed to individual point solutions) questions arise such as:

- How to use concepts such as stateless functions, actor models and RMI in actual programs, i.e., what are minimal/ideal bindings or extensions to programming languages so that programmers can take advantage of Computing in the Network?

- Are there additional, potentially higher-layer, abstractions that are needed/useful, for example data set synchronization, data types for distributed computing such as CRDTs?

In addition to programming languages, bindings, and data types, there is the question of execution environments and mobile code representation. With the vast number of different platforms (CPUs, GPUs, FPGAs etc.) it does not seem useful to assume exactly one environment. Instead, interesting applications might actually benefit from running one particular function on a highly optimized platform but are agnostic with respect to platforms for other, less performance-critical functions. Being able to support a heterogenous, evolving set of execution environments brings about questions such as:

- How to discover available platforms (and understand their properties)?

- How to specify application needs and map them to available platforms?

- Can a certain function/application service be provided with different fidelity levels, e.g., can an application leverage a GPU platform if available and fall back to a reduced feature set in case such a platform is not available?

In this context, updates and versioning could entail another dimension of variability for Computing in the Network:

- How to manage coexistence of multiple versions of functions and services, also for service routing and request forwarding?

- Is there potential for fallback and version negotiation if needed (considering the risk of "bidding downs" attacks?)
- How to retire old versions?
- How to securely and reliably deal with function updates and corresponding maintenance tasks?

5.5. Security, Privacy, Trust Model

Computing in the Network has interesting security-related challenges, including:

- How can a caller trust that a remote function works as expected? This entails several questions such as
  - How to securely bind "function names" to actual function code?
  - How to trust the execution platform (in its entirety)?
  - How to trust the network that is forwards requests (and result messages) reliably and securely?
  - How to ascertain that a function does what it claims to do?
- What levels of authentication are needed for callers (assuming that not everybody can invoke any function)?
- How to authenticate and achieve confidentiality for requests, their parameters and result data (especially when considering sharing of results)?

Many of these questions are related to other design decisions such as

- What kind of session concept do we assume, i.e., is there a concept of distributed application session that represents a trust domain for its members?
- Where is trust anchored? Can the system enable decentralized operation?

All of these questions are not new, but conceiving networking and computing holistically seems to revisit distributed systems and network security - because some established concepts and technologies may not be directly applicable (such as transport layer security and corresponding web PKI).

5.6. Coordination

For distributed systems, coordination is a key function and involves several functions such as configuration management, service discovery, application state management, and consensus schemes.

How these functions are implemented depends a lot on the nature of specific systems. For example, Apache ZooKeeper [7] is a logically centralized coordination service that provides coordination primitives to client application modules. The ZooKeeper itself is implemented as a distributed system consisting of a set of tightly coupled server instances that replicate the ZooKeeper state.

Other systems, such as the ICN-based CFN (Section 4.1) implement these services in a distributed way, employing different mechanisms for synchronization and consensus building.

While the fundamental concepts and mechanisms for coordination services are well understood, applying these concepts and mechanisms to a specific system design requires careful consideration.

5.7. Fault Tolerance, Failure Handling, Debugging, Management

Distributed computing naturally provides different types of failures and exceptions. In fine-granular distributed computing, some failures may by more tolerable (think microservices), i.e., platform crash or function abort due to isolated problems could be handled by just re-starting/re-running a particular function. Similarly, "message loss" or incorrect routing information may be repairable by the system itself (after time).

When failure cannot be repaired (or just tolerated) by the distributed computing framework, this raises questions such as:

- What are strategies for retrying vs aborting function invocation?
- How to signal exceptions and enable robust response to failures?

Failure handling and debugging also has a management aspect that leads to questions such as:

- What monitoring and instrumentation interfaces are needed?
- How can we represent, visualize, and understand the (dynamically changing) properties of Computing in the Network infrastructure as well as of the currently running/instantiated entities?

6. Acknowledgements

The authors would like to thank Dave Oran, Michal Krol, Spyridon Mastorakis, Yiannis Psaras, Eve Schooler, Dirk Trossen, and Phil Eardley for previous fruitful discussions on Computing in the Network topics and for feedback on this draft.

7. ChangeLog

7.1. 02

- fixed errors and updates references
- new Section 5.6 on Coordination
- renamed Section 5.7 to Fault Tolerance, Failure Handling, Debugging, Management
- new Section 4.2 on Akka in Section 4

7.2. 01

- added explanation of MEC and network slicing in Section 3.
- added clarification that edge computing is not limited to MEC
- added description of named service function chaining
- new Section 4 with a description of CFN-ICN

8. References

8.1. Informative References

[ACTIVE] Tennenhouse, D. and D. Wetherall, "Towards an active network architecture", ACM SIGCOMM Computer Communication Review Vol. 26, pp. 5-17, DOI 10.1145/231699.231701, April 1996.

[CANARY] Qu et al, H., "Canary -- A scheduling architecture for high performance cloud computing", 2016, <https://arxiv.org/abs/1602.01412>.

[CFN] KrA^3l, M., Mastorakis, S., Oran, D., and D. Kutscher, "Compute First Networking", Proceedings of the 6th ACM Conference on Information-Centric Networking, DOI 10.1145/3357150.3357395, September 2019.

[CROWD] Murray, D., Yoneki, E., Crowcroft, J., and S. Hand, "The case for crowd computing", Proceedings of the second ACM SIGCOMM workshop on Networking, systems, and applications on mobile handhelds - MobiHeld '10, DOI 10.1145/1851322.1851334, 2010.

[EDGESURVEY] Mach et al, P., "Mobile Edge Computing -- A Survey on Architecture and Computation Offloading", 2017, <https://ieeexplore.ieee.org/document/7879258>.

[FLINK] Katsifodimos, A. and S. Schelter, "Apache Flink: Stream Analytics at Scale", 2016 IEEE International Conference on Cloud Engineering Workshop (IC2EW), DOI 10.1109/ic2ew.2016.56, April 2016.

[FOGEDGE] Salaht, F., Desprez, F., and A. Lebre, "An Overview of Service Placement Problem in Fog and Edge Computing", ACM Computing Surveys Vol. 53, pp. 1-35, DOI 10.1145/3391196, July 2020.

[I-D.trossen-sfc-name-based-sff] Trossen, D., Purkayastha, D., and A. Rahman, "Name-Based Service Function Forwarder (nSFF) component within SFC framework", draft-trossen-sfc-name-based-sff-07 (work in progress), May 2019.

[MEC] ETSI, ., "Multi-access Edge Computing (MEC)", 2020, <https://www.etsi.org/technologies/multi-access-edge-computing>.

[MEC5G] Sami Kekki et al, ., "MEC in 5G Networks", 2018, <https://www.etsi.org/images/files/ETSIWhitePapers/etsi_wp28_mec_in_5G_FINAL.pdf>.

[MIST] Barik, R., Dubey, A., Tripathi, A., Pratik, T., Sasane, S., Lenka, R., Dubey, H., Mankodiya, K., and V. Kumar, "Mist Data: Leveraging Mist Computing for Secure and Scalable Architecture for Smart and Connected Health", Procedia Computer Science Vol. 125, pp. 647-653, DOI 10.1016/j.procs.2017.12.083, 2018.

[RAY] Moritz et al, P., "Ray -- A Distributed Framework for Emerging AI Applications", 2018, <http://dl.acm.org/citation.cfm?id=3291168.3291210>.

[RFC7665] Halpern, J., Ed. and C. Pignataro, Ed., "Service Function Chaining (SFC) Architecture", RFC 7665, DOI 10.17487/RFC7665, October 2015, <https://www.rfc-editor.org/info/rfc7665>.

[RICE] KrA^3l, M., Habak, K., Oran, D., Kutscher, D., and I. Psaras, "RICE", Proceedings of the 5th ACM Conference on Information-Centric Networking, DOI 10.1145/3267955.3267956, September 2018.

[SAPIO] Sapio, A., Abdelaziz, I., Aldilaijan, A., Canini, M., and P. Kalnis, "In-Network Computation is a Dumb Idea Whose Time Has Come", Proceedings of the 16th ACM Workshop on Hot Topics in Networks, DOI 10.1145/3152434.3152461, November 2017.

[SPARROW] Ousterhout, K., Wendell, P., Zaharia, M., and I. Stoica, "Sparrow", Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles - SOSP '13, DOI 10.1145/2517349.2522716, 2013.

8.2. URIs

[1] https://akka.io/

[2] https://flink.apache.org/

[3] https://doc.akka.io/docs/akka/2.3/scala/remoting.html

[4] https://doc.akka.io/docs/akka/current/general/addressing.html

[5] https://doc.akka.io/docs/akka/current/typed/routers.html

[6] https://doc.akka.io/docs/akka/current/typed/cluster.html

[7] https://zookeeper.apache.org/

Authors' Addresses

Dirk Kutscher
University of Applied Sciences Emden/Leer
Constantiaplatz 4
Emden D-26723
Germany

Email: ietf@dkutscher.net

Teemu Kaerkkaeinen
Technical University Muenchen
Boltzmannstrasse 3
Munich
Germany

Email: kaerkkae@in.tum.de

Joerg Ott
Technical University Muenchen
Boltzmannstrasse 3
Munich
Germany

Email: jo@in.tum.de

Computing in Network Research Group P. Liu
Internet-Draft L. Geng
Intended status: Informational China Mobile
Expires: January 13, 2021 July 12, 2020

Requirement of Computing in network
draft-liu-coinrg-requirement-03

Abstract

New technology such as IOT, edge computing,etc. propose the requirement of computing in network, so the convergence of network and computing has become a trend. It will bring some new directions and areas to be considered, such as the relationship between network and computing, the influence of integrating computing to the network, and so on.

This document points out the requirements of computing in network according to the development of new Industry, including the network and computing requirements.

Requirements Language

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].

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 13, 2021.

Table of Contents

1. Overview

The new services' provider expects a user experience with lower latency and high reliability, which put forwards immense challenges to cloud computing and traditional network. Centralized computing requires a long transmission distance of traffic flow, and the existing network technology is to the best of its ability. Network operators start to think about how to meet the higher needs of service provider and users. Computing in the network may solve the

problems because it can provide a flexible network and computing integration system.

To integrate the computing resource to the network, it need to find suitable computing nodes to handle service's request, as well as a forwarding path to them. How much computing resources will affect the delay of service processing, which could also affect the whole network latency. Just as the measurement of network performance has one more dimension, it will interact and cooperate with others. So there are some requirments for both network and computing.

2. Requirements of Network

The network requirements includes precision, concurrent, addressing and information interaction.

2.1. Precision

Precision of the network refers to the deterministic of latency, packet loss rate and perception of computing resources.

* Latency: The traditional network's best-effort forwarding mode can no longer meet the demand of such services for network latency. The deterministic latency brings forward a new measure latitude for network, which changes from in-time to on-time.

* Packet loss rate: It is another factor to evaluate the precision of the network. Utilizing the ubiquitous computing capability of the network, network prediction and segment-by-segment path retransmitting are realized based on AI, network transmission can be optimized and service QoS can be ensured.

* Perception of computing resources: how to precisely obtain the status of computing resource to meet the requirements of business requests is also a challenge to the network. It considers the network status and the performance status of computing resources can be matched dynamicly. So the user experience, utilization rate of computing resources and the network efficiency can be optimum.

For the latency and packet loss rate, some technologies such as time-sensitive network TSN, deterministic network DetNet, etc., have proposed corresponding technical means to provide network bearers with deterministic latency(IEEE802.1Qbv, IEEE802.1Qbu) and packet loss rate and guarantee the user's business experience. However, it also needs to consider how to guarantee the service's end-to-end latency, packet loss rate and resource utilization rate.

For the perception of computing resources, we may consider about the OAM and telemetry to achieve it, however, the performance and information collection strategies are issues that need attention.

2.2. Concurrent

There will be number of computing nodes deployed in the network, or computing fucntions intergrated in the network device for network computing. A serivce's computing request may distributed in several computing nodes in order to response quickly to the client. So there may be a lot of parallel computing task, whitch cause too much connection among the nodes but consume less bandwidth. It will bring great challenges to the concurrent network connection including how to build and deploy these distributed computing nodes to ensure the processing capability of the network , as well as the storage, call of the database are worth studying.

2.3. Addressing

Traditional application-based addressing can not accurately grasp the network performance in real time. The comprehensive performance of addressing results based on application layer may not be the best. It is always to find the consistent host's address and go through a long distance internet, which results in poor business experience.

It need to find some new way to improve the addressing proccess. For example, in the function based addressing, the application deconstruction components on the server side are distributed on the cloud platform, and the business logic in the server is transferred to the client side. So the client only needs to care about the computing function itself, not about the computing resources such as server, virtual machine, container and so on.

2.4. Information interaction

The network needs to have the ability to sense application's requirments and expose network and computing status. For example, application can tell the network requirements including bandwidth, latency and jitter, as well as the computing requirements, such as CPU, storage and memory. The network also can have the capability to be aware of the application's requirements. Thus it can effectively support the network programming, which could meet the future business requirements.

3. Requirements of computing

The computing requirements includes computing resource deployment, discovery, reservation, scheduling and OAM.

3.1. Computing resource deployment

If some computing tasks in the network is planned to be implemented, it needs to consider about what kinds of chips and where should them be deployed. On the one hands, different kinds of computing require different kinds of chips, such as CPU, GPU and memory chip. On the other hands, those chips may be put into router, switch, server or some dedicated machines, which are connected by the network.

There is an example about AI algorithm which might be discussed before. The AI algorithm has several steps including training and matching, and they also have different requirements of chips. In network computing, those steps could be distributed in different computing nodes.

3.2. Computing resource discovery

The network needs to have the ability to discover computing resources. when the computing nodes are deployed in the network, it need to be registered to the network management system, and the information of computing resource or routing can update. In this way, when there are computing tasks to be executed, the network can reasonably allocate resources according to the needs of the application.

3.3. Computing resource reservation

There might be serial distributed computing model of computing in the network, and different resources need to be reserved for different nodes. For example, AI algorithm now has a model of step-by-step iteration at multiple nodes. The previous iteration will affect the next calculation results, and the computing resources required for each iteration are not the same. From the perspective of network standard, we hope to regard computing resources as the dimensions to measure network performance, such as the same bandwidth, path, etc., while the traditional technologies of resource reservation have not considered the reservation of computing resources, and have not considered the differentiated resource reservation model. Therefore, new protocol or extension of existing protocol is needed to meet the requirement.

3.4. Computing aware scheduling

Computing in network needs a reasonable scheduling strategy, which means computing aware scheduling. According to the business requests, dynamicly computing power matching is carried out based on network status and performance of computing resources to achieve optimal user experience, optimal utilization of computing resources and optimal network efficiency. In computing aware scheduling, computing is seen as "link state" and the computing resource information should be exposed.

3.5. Computing resource OAM

The ability of OAM can be used to continuously update the current computing power resources, and perform some troubleshooting tasks. However, OAM of computing resource is more complex than network. Network monitoring is relatively simple, like bandwidth, latency, jitter, while computing can be divided into many categories, different application need different kinds of computing. So it need to implement fine-grained OAM of computing resource.

4. Requirements of management

The management requirements includes cross domain management, joint optimisation and multi user access.

4.1. Cross domain management

The computing in the network should ensure the end-to-end network management to meet the needs of different network topology, performance and function, which involves cross domain network arrangement. In the process of network data transmission, different services will forward in different ways or different network protocols, and computing resources may be distributed in different network domains. Effective cross domain management will enhance the performance of network and computing.

4.2. Joint optimisation

As computing resources are integrated in the network, and may be used as one of the measurement dimensions of network performance, joint optimization is also a very important part. Network and computing resources will affect each other, including performance, scheduling and so on. So It need a good joint optimization scheduling strategy.

4.3. Multi user access

Many existing applications, such as games, remote video conferencing, are usually multi--accessed and interacted by several users at the same time. This brings about the problem of service consistency, that is, users accessing to the same game or video need the consistency of SLA, otherwise it will seriously affect the experience of other users. Service consistency can be achieved through network management or application layer control.

5. Conclusion

Based on the requirements of new business, this document puts forward the requirements of computing in network, and gives some reference technologies and use cases. Computing in network is a new direction, some details need more in-depth discussion and research.

6. Security Considerations

Computing In network has brought the trend of network convergence in different regions. For example, 5g network of operators can go deep into the vertical industry user site to provide users with higher quality network services, which will bring the convergence of operator's network and user site network. Besides, industrial Internet brings the trend of integration of industrial OT network and IT network to further improve the production efficiency of the industry. It need to ensure the security of the network, including the mutual trust and non aggression of information among regions, which may require further protection and detection measures.

7. IANA Considerations

TBD.

8. Normative References

[I-D.kutscher-coinrg-dir]
Kutscher, D., Karkkainen, T., and J. Ott, "Directions for Computing in the Network", draft-kutscher-coinrg-dir-01 (work in progress), November 2019.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

Authors' Addresses

Peng Liu
China Mobile
Beijing 100053
China

Email: liupengyjy@chinamobile.com

Liang Geng
China Mobile
Beijing 100053
China

Email: gengliang@chinamobile.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 11, 2021

M. McBride
Futurewei
D. Kutscher
Emden University
E. Schooler
Intel
CJ. Bernardos
UC3M
D. Lopez
Telefonica I+D

July 10, 2020

Data Discovery Problem Statement
draft-mcbride-data-discovery-problem-statement-00

Abstract

If data is the new oil of the 21st century, then we need a standardized way of locating, capturing, classifying and transforming this raw data to generate insights and recommendations. Data, like oil, needs to be discovered and captured in order to be refined and valuable. While the topic of data discovery can be far reaching, this document focuses on the problem of actually locating data, throughout a network of data servers, in a standardized way.

Table of Contents

1. Introduction

There are myriad proprietary and standardized ways of discovering networking devices and hosts. There are many solutions for discovering data within a database. There are proprietary, non-standardized, ways of discovering the data that may be stored throughout an environment of networking devices. We can discover information about the devices but can't locate and capture stored data in a standard way. With more networking devices storing collected data there needs to be a standard way of discovering the specific data needed amongst a potentially huge lake of databases.

1.1. Requirements Language

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].

2. Problem Scope

Data may be cached, copied and/or stored at multiple locations in the network on route to its final destination. With an increasing percentage of devices connecting to the Internet being mobile,

support for in-the-network caching and replication is critical for continuous data availability. There are data repositories throughout a modern network and there needs to be a standardized way to locating the repositories and discovering the desired data within.

There are many types of relational (SQL) and non-relational (NoSQL) data classification solutions. Existing database classification engines allow for scanning of a database. We are defining the problem, however, of having a standards based solution to discover first where the databases exist throughout a network and then where specific data objects are located.

Data discovery is likely to look different depending on if we are seeking global vs local discovery. Data discovery may be location-driven. A standard to find data may want to search for it in a more proximal fashion, i.e., find the data that matches the search that is nearest to a location.

There is so much data being created, processed, and migrated, that it may only sometimes get stored more permanently in a database. There is going to be slightly less permanent data that resides for a time in memory, so that it may be discovered and accessed quickly. It may be more dynamic and short lived. Although we refer to the data store as a database, it may reside entirely in memory, and/or it may be stored in some other non-SQL indexing technology.

Each database essentially provides a directory service for the data within them and that directory service can be viewed as metadata. There is the need to understand where the databases/data lakes/ pockets of data reside. The location of each data store is the first level discovery problem, and the details of the database's directory is the second level discovery problem.

Publish and subscribe approaches allow nodes to express their interest in specific pieces of data without knowing the location of the data. There might be sources of data to be discovered that might not produce the specific data desired by the subscribers (or not produce data with a specific format or frequency). The subscriber will want to find the publishers which send the desired data characteristics.

3. Existing Solutions

3.1. Proprietary

There are many existing proprietary database discovery solutions we can evaluate in order to understand what aspects we need to standardized. For instance there is IBM Cognos, Wipro Data Discovery

Platform (DDP), and Amazon Macie among many others. Macie, for instance, is a data security and data privacy service that uses machine learning and pattern matching to discover and protect data in AWS. The service allows you to define data types in order to discover and protect the data that may be unique to a use case.

3.2. Opensource

There are opensource data solutions such as from ScienceBase (https://sciencebase.usgs.gov/). The U.S. Geological Survey (USGS) is developing ScienceBase, an open source, collaborative, scientific data and information management platform. It provides current documentation about its structure, information model, services, directory and repository. sbtools uses an R (command line driven program used to find data within the platform) interface for ScienceBase.

Another solution is the Interplanetary File System (ipfs.io). IPFS is a distributed system for storing and accessing files, websites, applications, and data. IPFS is a peer-to-peer (p2p) storage network. Content is accessible through peers, located anywhere in the world, that might relay information, store it, or do both. IPFS knows how to find what you ask for via its content address, rather than its location. There are three fundamental principles to understanding IPFS:

- Unique identification via content addressing
- Content linking via directed acyclic graphs (DAGs)
- Content discovery via distributed hash tables (DHTs)

4. Use Cases

Here are some of the use cases which will benefit from standards based data discovery solutions:

- We need a standards based solution to discover the increasing amount of data being stored in various locations throughout a network including at the edge. We need a standard protocol set for doing this data discovery, on the device or infrastructure edge, in order to meet the requirements of many use cases. We will have terabytes of data on the edge and need a way to identify its existence and find the desired data. [I-D.mcbride-edge-data-discovery-overview] is focusing on this aspect of data discovery.

- We need a secure standards based solution for data discovery. Several of the proprietary secure data discovery solutions use machine learning and pattern matching to discover and protect the data. We need to incorporate existing, or new, ietf security solutions when discoverying data.

- We need a standards based solution for using named based solutions for data discovery. An Information Centric Networking (ICN) enabled network routes data by name (vs address), caches content natively in the network, and employs data-centric security. Data discovery may require that data be associated with a name or names, a series of descriptive attributes, and/or a unique identifier. NDN (Named Data Networking) can be applied to edge data discovery to make it much easier to extract data and meta-data by naming it. If data was named we would be able to discover the appropriate data simply by its name.

- We need a standards based way of discovering data in mobile wireless networks. Data could reside on the eNodeB or other wireless access infrastructure equipment in addition to residing on servers in the packet core.

5. IANA Considerations

N/A

6. Security Considerations

Data and metadata discovery are both a function of who asks for the data and in what context. The policies attached to the database and the metadata are going to dictate what view into the data that the system returns to the requester.

7. Acknowledgement

8. Normative References

[I-D.mcbride-edge-data-discovery-overview]
McBride, M., Kutscher, D., Schooler, E., and C. Bernardos, "Edge Data Discovery for COIN", draft-mcbride-edge-data-discovery-overview-03 (work in progress), January 2020.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

Authors' Addresses

Mike McBride
Futurewei

Email: michael.mcbride@futurewei.com

Dirk Kutscher
Emden University

Email: ietf@dkutscher.net

Eve Schooler
Intel

Email: eve.m.schooler@intel.com
URI: http://www.eveschooler.com

Carlos J. Bernardos
Universidad Carlos III de Madrid
Av. Universidad, 30
Leganes, Madrid 28911
Spain

Phone: +34 91624 6236
Email: cjbc@it.uc3m.es
URI: http://www.it.uc3m.es/cjbc/

Diego R. Lopez
Telefonica I+D
Don Ramon de la Cruz, 82
Madrid 28006
Spain

Email: diego.r.lopez@telefonica.com

COINRG
Internet-Draft
Intended status: Standards Track
Expires: January 14, 2021

M. McBride
Futurewei
D. Kutscher
Emden University
E. Schooler
Intel
CJ. Bernardos
UC3M
D. Lopez
Telefonica
July 13, 2020

Edge Data Discovery for COIN
draft-mcbride-edge-data-discovery-overview-04

Abstract

This document describes the problem of distributed data discovery in edge computing, and in particular for computing-in-the-network (COIN), which may require both the marshalling of data at the outset of a computation and the persistence of the resultant data after the computation. Although the data might originate at the network edge, as more and more distributed data is created, processed, and stored, it becomes increasingly dispersed throughout the network. There needs to be a standard way to find it. New and existing protocols will need to be developed to support distributed data discovery at the network edge and beyond.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 14, 2021.

Table of Contents

1. Introduction

Edge computing is an architectural shift that migrates Cloud functionality (compute, storage, networking, control, data management, etc.) out of the back-end data center to be more proximate to the IoT data being generated and analyzed at the edges of the network. Edge computing provides local compute, storage and

connectivity services, often required for latency- and bandwidth-sensitive applications. Thus, Edge Computing plays a key role in verticals such as Energy, Manufacturing, Automotive, Video Surveillance, Retail, Gaming, Healthcare, Mining, Buildings and Smart Cities.

1.1. Edge Data

Edge computing is motivated at least in part by the sheer volume of data that is being created by endpoint devices (sensors, cameras, lights, vehicles, drones, wearables, etc.) at the very network edge and that flows upstream, in a direction for which the network was not originally designed. In fact, in dense IoT deployments (e.g., many video cameras are streaming high definition video), where multiple data flows collect or converge at edge nodes, data is likely to need transformation (to be transcoded, subsampled, compressed, analyzed, annotated, combined, aggregated, etc.) to fit over the next hop link, or even to fit in memory or storage. Note also that the act of performing computation on the data creates yet another new data stream! Preservation of the original data streams is needed sometimes but not always.

In addition, data may be cached, copied and/or stored at multiple locations in the network on route to its final destination. With an increasing percentage of devices connecting to the Internet being mobile, support for in-the-network caching and replication is critical for continuous data availability, not to mention efficient network and battery usage for endpoint devices.

Additionally, as mobile devices' memory/storage fill up, in an edge context they may have the ability to offload their data to other proximate devices or resources, leaving a bread crumb trail of data in their wakes. Therefore, although data might originate at edge devices, as more and more data is continuously created, processed and stored, it becomes increasingly dispersed throughout the physical world (outside of or scattered across managed local data centers), increasingly isolated in separate local edge clouds or data silos. Thus, there needs to be a standard way to find it. New and existing protocols will need to be identified/developed/enhanced for these purposes. Being able to discover distributed data at the edge or in the middle of the network will be an important component of Edge computing.

1.2. Background

Several IETF T2T RG Edge Computing discussions have been held over the last couple years. A comparative study on the definition of Edge computing was presented in multiple sessions in T2T RG in 2018 and an

Edge Computing I-D was submitted early 2019. An IETF BEC (beyond edge computing) effort has been evaluating potential gaps in existing edge computing architectures. Edge Data Discovery is one potential gap that was identified and that needs evaluation and a solution. The newly proposed COIN RG highlights the need for computations in the network to be able to marshal potentially distributed input data and to handle resultant output data, i.e., its placement, storage and/or possible migration strategy.

1.3. Requirements Language

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].

1.4. Terminology

- Edge: The edge encompasses all entities not in the back-end cloud. The device edge represents the very leaves of the network and encompasses the entities found in the last mile network. Sensors, gateways, compute nodes are included. Because the things that populate the IoT can be both physical and/or cyber, in some solutions, particularly in software-defined or digital-twin contexts, the device edge can include logical (vs physical) entities. The infrastructure edge includes equipment on the network operator side of the last mile network including cell towers, edge data centers, cable headends, POPs, etc. See Figure 1 for other possible tiers of edge clouds between the device edge and the back-end cloud data center.

- Edge Computing: Distributed computation that is performed near the network edge, where nearness is determined by the system requirements. This includes high performance compute, storage and network equipment on either the device or infrastructure edge.

- Edge Data Discovery: The process of finding required data from edge entities, i.e., from databases, file systems, and device memory that might be physically distributed in the network, and providing access to it logically as if it were a single unified source, perhaps through its namespace, that can be evaluated or searched.

- ICN: Information Centric Networking. An ICN-enabled network routes data by name (vs address), caches content natively in the network, and employs data-centric security. Data discovery may require that data be associated with a name or names, a series of descriptive attributes, and/or a unique identifier.

2. Edge Data Discovery Problem Scope

Our focus is on how to define and scope the edge data discovery problem. This requires some discussion of the evolving definition of the edge as part of a cloud-to-edge continuum and in turn what is meant by edge data, as well as the meta-data about the edge data.

2.1. A Cloud-Edge Continuum

Although Edge Computing data typically originates at edge devices, there is nothing that precludes edge data from being created anywhere in the cloud-to-edge computing continuum (Figure 1). New edge data may result as a byproduct of computation being performed on the data stream anywhere along its path in the network. For example, infrastructure edges may create new edge data when multiple data streams converge upon this aggregation point and require transformation (e.g., to fit within the available resources, to smooth raw measurements to eliminate high-frequency noise, or to obfuscate data for privacy).

Initially our focus is on discovery of edge data that resides at the Device Edge and the Infrastructure Edge.

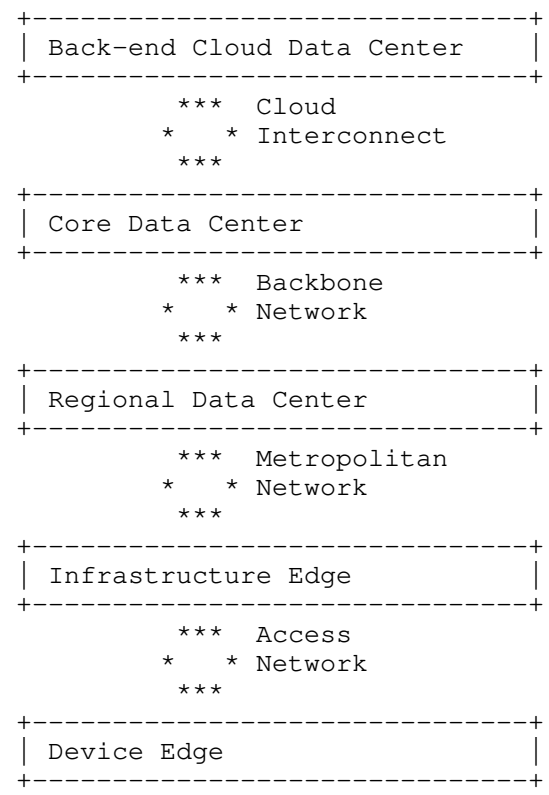

Figure 1: Cloud-to-edge computing continuum

2.2. Types of Edge Data

Besides classically constrained IoT device sensor and measurement data accumulating throughout the edge computing infrastructure, edge data may also take the form of higher frequency and higher volume streaming data (from a continuous sensor or from a camera), meta data (about the data), control data (regarding an event that was triggered), and/or an executable that embodies a function, service, or any other piece of code or algorithm. Edge data also could be created after multiple streams converge at an edge node and are processed, transformed, or aggregated together in some manner.

Regardless of edge data type, a key problem in the Cloud-Edge continuum is that data is often kept in silos. Meaning, data is often sequestered within the Edge where it was created. A goal of this discussion is to consider the prospect that different types of edge data will be made accessible across disparate edges, for example to enable richer multi-modal analytics. But this will happen only if

data can be described, searched and discovered across heterogeneous edges in a standard way. Having a mechanism to enable granular edge data discovery is the problem that needs solving either with existing or new protocols. The mechanisms shouldn't care to which flavor cloud or edge the request for data discovery is made.

3. Edge Scenarios Requiring Data Discovery

1. A set of data resources appears (e.g., a mobile node hosting data joins a network) and they want to be discoverable by an existing but possibly virtualized and/or ephemeral data directory infrastructure.

2. A device wants to discover data resources available at or near its current location. As some of these resources may be mobile, the available set of edge data may vary over time.

3. A device wants to discover to where best in the edge infrastructure to opportunistically upload its data, for example if a mobile device wants to offload its data to the infrastructure (for greater data availability, battery savings, etc.).

4. Edge Data Discovery

How can we discover data on the edge and make use of it? There are proprietary implementations that collect data from various databases and consolidate it for evaluation. We need a standard protocol set for doing this data discovery, on the device or infrastructure edge, in order to meet the requirements of many use cases. We will have terabytes of data on the edge and need a way to identify its existence and find the desired data. A user requires the need to search for specific data in a data set and evaluate it using their own tools. The tools are outside the scope of this document, but the discovery of that data is in scope.

4.1. Types of Discovery

There are many aspects of discovery and many different protocols that address each aspect.

Discovery of new devices added to an environment. Discovery of their capabilities/services in client/server environments. Discovery of these new devices automatically. Discovering a device and then synchronizing the device inventory and configuration for edge services. There are many existing protocols to help in this discovery: UPnP, mDNS, DNS-SD, SSDP, NFC, XMPP, W3C network service discovery, etc.

Edge devices discover each other in a standard way. We can use DHCP, SNMP, SMS, COAP, LLDP, and routing protocols such as OSPF for devices to discover one another.

Discovery of link state and traffic engineering data/services by external devices. BGP-LS is one such solution.

The question is if one or more of these protocols might be a suitable contender to extend to support edge data discovery?

4.2. Naming the Data

Information-Centric Networking (ICN) RFC 7927 [RFC7927] is a class of architectures and protocols that provide "access to named data" as a first-order network service. Instead of host-to-host communication as in IP networks, ICNs often use location-independent names to identify data objects, and the network provides the services of processing (answering) requests for named data with the objective to finally deliver the requested data objects to a requesting consumer.

Such an approach has profound effects on various aspects of a networking system, including security (by enabling object-based security on a message/packet level), forwarding behavior (name-based forwarding, caching), but also on more operational aspects such as bootstrapping, discovery etc.

The CCNx and NDN (https://named-data.net) variants of ICN are based on a request/response abstraction where consumers (hosts, application requesting named data) send INTEREST messages into the network that are forwarded by network elements to a destination that can provide the requested named data object. Corresponding responses are sent as so-called DATA messages that follow the reverse INTEREST path.

Each unique data object is named unambiguously in a hierarchical naming scheme and is authenticated through Public-Key cryptography (data objects can also optionally be encrypted in different ways). The naming concept and the object-based security approach lay the foundation for location-independent operation. The network can generally operate without any notion of location, and nodes (consumers, forwarders) can forward requests for named data objects directly, i.e., without any additional address resolution. Location independence also enables additional features, for example the possibility to replicate and cache named data objects. On-path caching is a standard feature in many ICN systems -- typically for enhancing reliability and performance.

In CCNx and NDN, forwarders are stateful, i.e., they keep track of forwarded INTEREST to later match the received DATA messages.

Stateful forwarding (in conjunction with the general named-based and location-independent operation) also empowers forwarders to execute individual forwarding strategies and perform optimizations such as in-network retransmissions, multicasting requests (in cases there are several opportunities for accessing a particular named data object) etc.

Naming data and application-specific naming conventions are naturally important aspects in ICN. It is common that applications define their own naming convention (i.e., semantics of elements in the name hierarchy). Such names can often be directly derived from application requirements, for example a name like /my-home/living-room/light/switch/main could be relevant in a smart home setting, and corresponding devices and applications could use a corresponding convention to facilitate controllers finding sensors and actors in such a system with minimal user configuration.

The aforementioned features make ICN amenable to data discovery. Because there is no name/address chasm as in IP-based systems, data can be discovered by sending an INTEREST to named data objects directly (assuming a naming convention as described above). Moreover, ICN can authenticate received data objects directly, for example using local trust anchors in the network (for example in a home network).

Advanced ICN features for data discovery include the concept of manifests in CCNx, i.e., ICN objects that describe data collections, and data set synchronization protocols in NDN (https://named-data.net/publications/li2018sync-intro/) that can inform consumers about the availability of new data in a tree-based data structure (with automatic retrieval and authentication). Also, ICN is not limited to accessing static data. Frameworks such as Named Function Networking (http://www.named-function.net) and RICE can provide the general ICN feature for discovery not only for data but also for name functions (for in-network computing) and for their results.

5. Use Cases of Edge Data Discovery

5.1. Autonomous Vehicles

Autonomous vehicles rely on the processing of huge amounts of complex data in real-time for fast and accurate decisions. These vehicles will rely on high performance compute, storage and network resources to process the volumes of data they produce in a low latency way. Various systems will need a standard way to discover the pertinent data for decision making.

5.2. Video Surveillance

The majority of the video surveillance footage will remain at the edge infrastructure (not sent to the cloud data center). This footage is coming from vehicles, factories, hotels, universities, farms, etc. Much of the video footage will not be interesting to those evaluating the data. A mechanism, perhaps a set of protocols, is needed to identify the interesting data at the edge. What constitutes interesting will be context specific, e.g., a video frame might be considered interesting if and only if it includes a car, or person, or bicyclist, or a backyard nocturnal creature, or etc. Interesting video data may be stored longer in storage systems at the very edge of the network and/or in networking equipment further away from the device edge that has access to data in flight as it transits the network.

5.3. Elevator Networks

Elevators are one of many industrial applications of edge computing. Edge equipment receives data from hundreds of elevator sensors. The data coming into the edge equipment is vibration, temperature, speed, level, video, etc. We need the ability to identify where the data we need to evalute is located.

5.4. Service Function Chaining

Service function chaining (SFC) allows the instantiation of an ordered set of service functions (SFs) and the subsequent "steering" of traffic through them. Service functions are expected to be deployed at the edge of the network, as a feasible deployment of "Compute In the Network", with multiple types of potential use cases (e.g., fog robotics, Industry 4.0 automation, etc). Service functions provide a specific treatment of received packets, therefore they need to be discoverable so they can be used in a given service composition via SFC. In addition, these functions can be producers and/or consumers of data. So far, how the functions are discovered and composed has been out of the scope of discussions in the IETF. While there are some mechanisms that can be used and/or extended to provide this functionality, more work needs to be done. An example of this can be found in [I-D.bernardos-sfc-discovery].

In an SFC environment deployed at the edge, the discovery protocol may also need the following kind of meta-data information per (service) function:

- Service Function Type: identifying the category of function provided.

- SFC-aware: Yes/No. Indicates if the function is SFC-aware.
- Route Distinguisher (RD): IP address indicating the location of the function.
- Pricing/costs details.
- Migration capabilities of the function: whether a given function can be moved to another provider (potentially including information about compatible providers topologically close).
- Mobility of the device hosting the function, with e.g. the following sub-options:

  Level: no, low, high; or a corresponding scale (e.g., 1 to 10).

  Current geographical area (e.g., GPS coordinates, post code).

  Target moving area (e.g., GPS coordinates, post code).
- Power source of the device hosting the function, with e.g. the following sub-options:

  Battery: Yes/No. If Yes, the following sub-options could be defined:

    Capacity of the battery (e.g., mmWh).

    Charge status (e.g., %).

    Lifetime (e.g., minutes).

Discovery of resources in an NFV environment: virtualized resources do not need to be limited to those available in traditional data centers, where the infrastructure is stable, static, typically homogeneous and managed by a single admin entity. Computational capabilities are becoming more and more ubiquitous, with terminal devices getting extremely powerful, as well as other types of devices that are close to the end users at the edge (e.g., vehicular onboard devices for infotainment, micro data centers deployed at the edge, etc.). It is envisioned that these devices would be able to offer storage, computing and networking resources to nearby network infrastructure, devices and things (the fog paradigm). These resources can be used to host functions, for example to offload/ complement other resources available at traditional data centers, but also to reduce the end-to-end latency or to provide access to specialized information (e.g., context available at the edge) or hardware. Similar to the discovery of functions, while there are

mechanisms that can be reused/extended, there is no complete solution yet defined. An example of work in this area is [I-D.bernardos-intarea-vim-discovery]. The availability of this meta-data about the capabilities of nearby physical as well as virtualized resources can be made discoverable through edge data discovery mechanisms.

5.5. Ubiquitous Witness

Ubiquitous Witness (UW) is the name of a use case that has been presented in past COINRG and ICNRG meetings at the IETF. It describes what might occur in dense IoT deployments when an anomaly occurs. There are many "witnesses" to report on what happened within a limited region of interest and around an approximate point in time. The use case highlights the need for upstream data discovery and management. It is agnostic to where the dense IoT deployment resides, whether in a factory, home, commercial building, city, entertainment venue, et cetera. For example, as cameras and other sensors have become ubiquitous in Smart Cities, it would be helpful to be able to discover and examine data from all devices and sensors that witnessed an accident in a city intersection; this could be data from cameras mounted at the intersection itself, on nearby buildings, in cars, and cell phones of individuals on location.

If an anomaly were to automatically trigger independent upstream flows of video data from all of the witnesses (within a proximal vicinity and time window), the data flows would naturally converge at shared collection or aggregation points in the network. These edge nodes might opt to vault any data deemed part of a safety-related anomaly, which would enable interested parties (the car owner, the car manufacturer, an insurance company, a city traffic planner) to investigate the root cause of the anomaly after the fact. The implication however is that enough meta data has been generated alongside the data itself (e.g., a data name, an identifier, or a geo location and timestamp), to allow the retrieval of this distributed data, provided those asking have proper authorization to access it.

The UW streams are contextually-related and as such it can be advantageous also to be able to process them simultaneously, at the time they are first generated. For example if collection nodes could derive that groups of data streams are contextually-related, they could stitch streams together to create a 360-degree view of the anomalous event (e.g., to walk around in the data), or to winnow the set of vaulted data to only the "best" video (e.g., highest resolution, unoccluded views) or to perform compute-in-the-network to enable them to fit within the available resources (e.g., at the receiving node due to the convergence or implosion of upstream data, or over the next hoplink). Ubiquitous Witness data doesn’t have to

be video data, but video illustrates why one might want to jointly process upstream flows in real-time.

6. IANA Considerations

N/A

7. Security Considerations

Security considerations will be a critical component of edge data discovery particularly as intelligence is moved to the extreme edge where data is to be extracted.

An assumption is that all data will have associated policies (default, inherited or configured) that describe access control permissions. Consequently, the discoverability of data will be a function of who or what has requested access. In other words, the discoverable view into the available data will be limited to those who are authorized. Discovering edge data that is exclusively private is out of scope of this document, the assumption being that there will be some edge clouds that do not expose or publish the availability of their data. Although edge data may be sent to the back-end cloud as needed, there is nothing that precludes it from being discoverable if the cloud offers it as public.

8. Acknowledgement

The authors thank Dave Oran for his detailed feedback on an early version of this draft, as well as inputs from Greg Skinner and Lixia Zhang.

9. Normative References

[I-D.bernardos-intarea-vim-discovery]
Bernardos, C. and A. Mourad, "IPv6-based discovery and association of Virtualization Infrastructure Manager (VIM) and Network Function Virtualization Orchestrator (NFVO)", draft-bernardos-intarea-vim-discovery-03 (work in progress), February 2020.

[I-D.bernardos-sfc-discovery]
Bernardos, C. and A. Mourad, "Service Function discovery in fog environments", draft-bernardos-sfc-discovery-04 (work in progress), March 2020.

[I-D.irtf-icnrg-ccnxmessages]
Mosko, M., Solis, I., and C. Wood, "CCNx Messages in TLV Format", draft-irtf-icnrg-ccnxmessages-09 (work in progress), January 2019.

[I-D.irtf-icnrg-ccnxsemantics]
Mosko, M., Solis, I., and C. Wood, "CCNx Semantics", draft-irtf-icnrg-ccnxsemantics-10 (work in progress), January 2019.

[I-D.kutscher-icnrg-rice]
Krol, M., Habak, K., Oran, D., Kutscher, D., and I. Psaras, "Remote Method Invocation in ICN", draft-kutscher-icnrg-rice-00 (work in progress), October 2018.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC7927] Kutscher, D., Ed., Eum, S., Pentikousis, K., Psaras, I., Corujo, D., Saucez, D., Schmidt, T., and M. Waehlisch, "Information-Centric Networking (ICN) Research Challenges", RFC 7927, DOI 10.17487/RFC7927, July 2016, <https://www.rfc-editor.org/info/rfc7927>.

Authors' Addresses

Mike McBride
Futurewei

Email: michael.mcbride@futurewei.com

Dirk Kutscher
Emden University

Email: ietf@dkutscher.net

Eve Schooler
Intel

Email: eve.m.schooler@intel.com
URI: http://www.eveschooler.com

Carlos J. Bernardos
Universidad Carlos III de Madrid
Av. Universidad, 30
Leganes, Madrid 28911
Spain

Phone: +34 91624 6236
Email: cjbc@it.uc3m.es
URI: http://www.it.uc3m.es/cjbc/

Diego R. Lopez
Telefonica

Email: diego.r.lopez@telefonica.com
URI: https://www.linkedin.com/in/dr2lopez/