Technical Disclosure Commons

Defensive Publications Series

April 2021

AN INTENT-BASED CACHING SYSTEM FOR CONSTRAINED INTERNET-OF-THINGS ENVIRONMENTS

Dima Dababneh

Sean Ceballos-McGee

Connie Sidberry

Robert Barton

AN INTENT-BASED CACHING SYSTEM FOR CONSTRAINED INTERNET-OF-THINGS ENVIRONMENTS

AUTHORS:
Dima Dababneh
Sean Ceballos-McGee
Connie Sidberry
Robert Barton

ABSTRACT

Internet-of-Things (IoT) environments often have very constrained edge devices, with limited memory, storage, and compute power. Additionally, these environments can be highly distributed. In many cases, edge devices/applications need to access content repeatedly from a cloud service or higher-tiered application. Because device and network connections can become constrained, it is desirable to cache objects, files, etc. local to the edge devices. However, caching systems and cache replacement algorithms may not consider the criticality or requirements of one application over another. Techniques presented herein provide for adapting well-known caching mechanisms for utilization within environments involving constrained networks. Further, intent-based classification techniques presented herein can facilitate preserving and caching critical data at network edges.

DETAILED DESCRIPTION

Many IoT environments are highly distributed and often include very constrained edge devices having limited memory, storage, and compute power. For example, a distributed water management system may utilize a cellular backhaul to communicate with IoT gateway devices. Increasingly, gateway devices at network edges are running applications that are distributed to the edge. These edge applications can provide a host of potential services including local trending and analytics, Artificial Intelligence (AI)/Machine Learning (ML) trained models, distributed Supervisory Control and Data Acquisition (SCADA) services, and many more.

In many cases, edge applications need to access content repeatedly from a cloud service or a higher-tiered application in a Fog architecture. Because device and network

connections are often constrained, it is desirable for locally available resources to cache "high-value" objects, files, or even byte-level blocks. By caching "high-value" data at an IoT gateway, performance at the edge can improve considerably, which is well-known by caching systems.

However, caching systems and cache replacement algorithms may not consider the criticality or requirements of one application over another. Most caching systems use other metrics like recency (for example, most/ least recently accessed) or popularity (for example, cache item hit count) when determining how to manage a cache. However, in IoT environments, even rarely accessed objects may require a much higher caching priority due to the latency requirements associated with the application when accessing the objects. In such cases, the pieces of data may be considered “high-value” and should be preserved at the gateway device, even if they are rarely accessed.

This proposal provides techniques to leverage Intent-Based Networking (IBN) as a mechanism for classifying the criticality of an application and then use that classification as an input for cache prioritization at the edge. The criticality of data to be cached can be determined by various key performance indicators (KPIs) such as (but not limited to):

- End-to-end (E2E) Latency requirements;
- Importance of the application (e.g., Is it mission-critical or best-effort?);and/or
- Revenue impact of losing the application or not meeting application Service Level Agreements (SLAs).

Such a latency-focused caching strategy can utilize the same mechanisms and hardware that are in place for network traffic prioritization. Figure 1, below, illustrates various example operations that can be utilized to achieve latency-focused caching in accordance with techniques of this proposal.

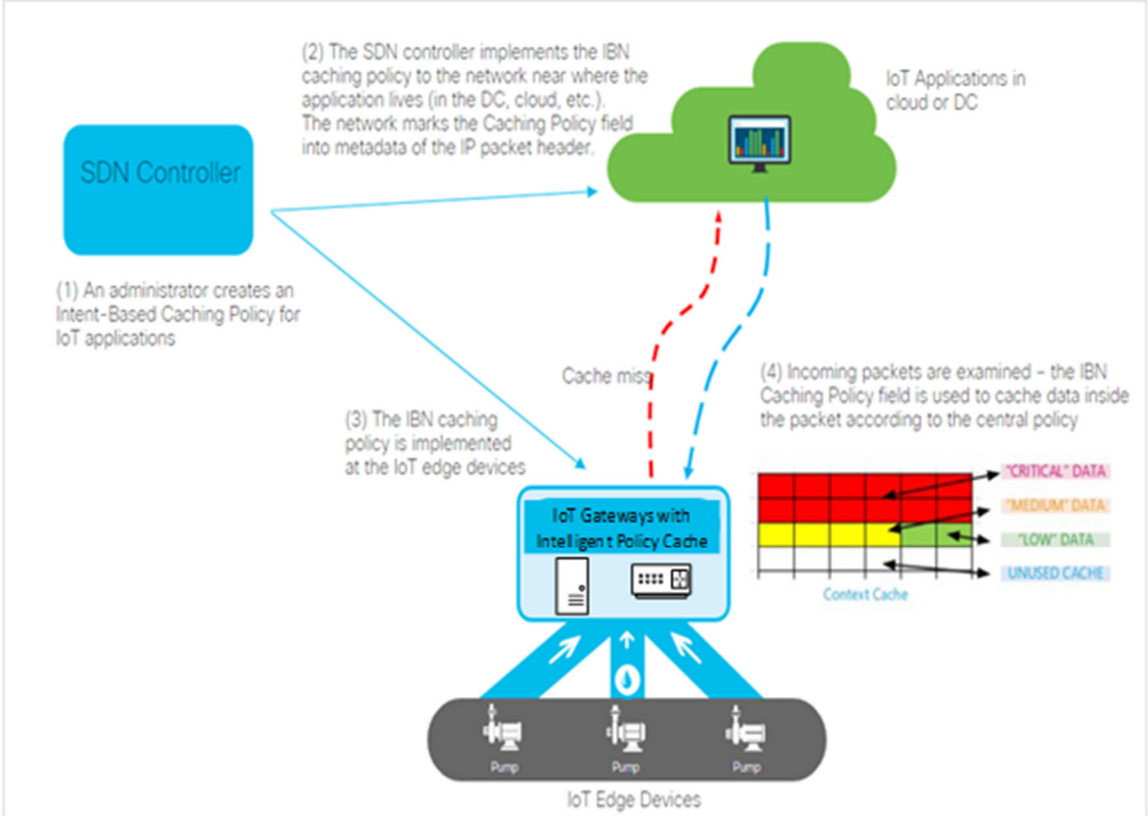

Figure 1: Example Latency-Focused Caching Operations

As illustrated in Figure 1, a first step involves introducing a centralized application policy tool and Software Defined Network (SDN) orchestration device, orchestrator, or controller. Today, such tools can be used to create Quality of Service (QoS) policies for the network. An application's intent can be expressed in the orchestration device. The intent can be translated to a QoS marking and provided to a management system by the network.

The SDN controller can centrally configure the caching intent of an IoT device or application according to the KPIs that are associated with performance at the edge (e.g., the KPIs noted above) and translate this into an edge caching policy.

For a second step, the network at the cloud service (or north-bound IoT application) implements the policy of the SDN controller. Traffic is analyzed by the cloud or central network elements. Once packets containing critical application contents are identified, the IBN caching policy is applied to these packets.

In general, a "Caching Policy" field can exist in the metadata of the packet header to be utilized such that an IoT edge or gateway device can interpret the priority to cache

content within a packet. By way of implementation, the Caching Policy can be written into Internet Protocol (IP) packet metadata such as within an IP version 6 (IPv6) Extension Header or the like.

In one instance, the Caching Policy field can be used to rank the criticality of the data. This ranking can be as simple or complex as the policy administrator requires. An example ranking may be characterized as:

- Critical;
- Medium;
- Low; or
- No Priority Associated

This caching policy could use metadata in the packet header to classify caching priorities in a similar manner as network hardware uses QoS traffic classes for differentiating between network traffic priorities.

For a third step, the SDN controller communicates with the edge devices and gateways and instructs them of the caching policy that is to be used. Because these devices have constrained storage capabilities, the caching policy may change on a per-location or per-context basis; thus, the IoT edge devices and gateways are to have a coordinated mechanism for managing their available (constrained) caching space as well as for translating the caching policy in the IP packet metadata field into an action.

In a fourth step at the edge, either the IoT device or an edge gateway accesses content from the cloud or another north-bound resource. Unfortunately, there may be a cache miss in some instances.

As packets are received by the IoT edge device, the Caching Policy metadata field is examined in the packet header and the edge device will update the local cache according to the configured policy. For example, one policy may stipulate that data (file, object, byte sequence) marked as Critical will be cached indefinitely. If any space remains on the edge device, Medium or Low classified objects may take the remaining space. The Critical data will be kept in the cache regardless of how many hits the object receives, ensuring that when it is next requested, the data will be locally available.

In another instance, a Caching Policy may be created that gives a time limit to the retention of cached objects. If they have not been accessed for some time the objects may

have their ranking changed. For example, if a Critical data object has not been accessed in the last X days, its priority can be downgraded to "Medium" to make way for other Critical data objects.

The techniques herein do not involve caching IP packets but rather web and/or JavaScript Object Notation (JSON) objects and other files that may be used by edge applications. As the nature and purpose of edge applications is likely to be greatly expanded beyond what is used today, it is expected that many of these applications will work in a hierarchical manner, where the edge will access central or cloud content through Application Programming Interfaces (APIs), typically Representation State Transfer (REST) and/or JSON formatted objects. As such, many of these objects will be cacheable, albeit on a constrained edge platform.

In some instances, the SDN controller that orchestrates intent-based marking of packet headers carrying an object/file may also provide details that indicate how the object/file should be cached, including a lifetime of the object/file. Thus, when the object or file is cached at the edge, an entry can be provided in the edge registry indicating the freshness frequency check of the object. It should be noted, as discussed above, that when data has not been retrieved for a set period of time it can either demoted in priority or flushed. According to this strategy, if an object/file is deemed critical, a freshness frequency can also be included in packet headers.

Further, it should be noted that QoS markings based on Differentiated Services Code Point (DSCP) define a Per-Hop Behavior (PHB) of an IP packet and have no concept of cache-ability. In other words, QoS alone may not suggest the need for object/file caching. For example, consider two applications that mark DSCP exactly the same, but have completely different caching requirements.

Thus, the approach described herein involving a central controller through which caching intent can be expressed on a per-application basis and marked directly into every packet header (a marking that is carried wherever a packet may travel) will likely be useful to the future of edge computing. By carrying caching instructions, the packet header for every object/file that is downloaded, technique herein intrinsically direct the edge to manage the caching of every object/file being sent. The beauty of this approach is that it can be centrally orchestrated and controlled by an administrator or application owner.

Further, techniques herein provide for classifying data in a tiered fashion (e.g., critical, medium, and low) across different applications and/or traffic streams. Over time, an object/file in a higher class that has not been accessed for a period of time can be demoted to a lower class and can eventually be flushed. Thus, the logic that is possible due to the cache-ability and priority metadata allows a wide variety of cache management possibilities. For example, in some instances, the concepts described herein can also be broadened to other areas such as 5G Mobile Edge Computing, etc.

In summary, techniques herein provide intent-based techniques through which application cache-ability requirements can be configured centrally through an SDN controller, thereby allowing applications with a higher need for object/file caching at the edge to be identified and given an appropriate caching priority. This, in turn, is translated to a metadata marking scheme for packet headers before the object/file is transmitted to the edge to facilitate edge caching according to caching priority.