CIMI Corporation

https://www.cimicorp.com/

email: services@cimicorp.com

Special Report: Edge Computing, Drivers, Technologies, Impacts

While there are surely many contenders for the most critical and hyped technology, edge computing is my choice. 5G depends on the edge, and so does IoT. Many believe it's the future of cloud computing. Many believe it's the future of premises computing. Everybody seems to believe that it will make them money, despite the fact that different definitions of "the edge" would foreclose the participation of some of the hopefuls.

For all this believing, though, we don't have much of a picture about what the edge has to be, and that's what I'm going to work to correct in a series of blogs. The starting point for this, and my focus here, is the specific issue set that edge computing has to address. I'll then develop the individual issues and how the market seems to be addressing them.

Edge Issues and Answers

The first issue for the edge is **what differentiates it**? There's not much sense in talking about edge computing if it looks just like other forms of computing, meaning that it has the same financial and technical properties. This is the starting-point issue, and the one we'll cover now.

Enterprises say that "edge computing" is computing located in "close proximity" to the point of activity the applications being run are supporting. If they're pressed for why that's important, they'll respond that some application missions require low latency, meaning that they require a quick response to a request. IoT is the example they'll usually offer. If pressed further, they'll say that it's likely that edge missions generate events rather than transactions, and that the applications have to be synchronized with real-world conditions. So let's say that edge computing is computing designed to process latency-sensitive events that couple directly to the real world.

The idea of close proximity implies some geographic placement; we have to know where event sources are in order to know what would be close to them. This introduces the concepts of "event geography", the distribution of the event sources in space, "event density", the concentration of event sources in a given geographical area, and "event source mobility", the extent to which an event source can change locations. These factors are critical in deciding whether edge computing is deployed on the premises or provided as a service.

Where event geography is concentrated, density is high, and mobility is minimal (a factory floor, a warehouse, etc.) we could expect to deploy edge facilities within the location where event sources are concentrated. That means on-premises hosting, likely not in the data center but in a location close to the event source concentration. This is a **local edge**.

The problem with local edge computing arises when one or more of our three event demographics changes. If event geography is large, if event sources are highly mobile, or if concentrations are low, local-edge processing may be impossible or expensive, in which case we need a different edge model. Just what model is required would depend on just how event demographics changed, and the critical question seems to be the event geography and mobility.

Given that access networks terminate within a metro complex, it seems likely that edge computing as a service would be available at the metro level. This assumption yields two edge-as-a-service models, the **metro-centric model** and the **distributed edge model**. Metro-centric edge hosting would support applications where event sources were dominantly located within a single metro area, and if they moved at all, moved dominantly within that area. Distributed-edge hosting then supports event sources

that are distributed beyond a metro, or move across multiple metro areas.

The final event processing characteristic we need to consider is “processing depth”, which is a measure of how far an event workflow extends. Note that this isn't a geographical concept but a workflow complexity concept, an indication of whether an event simply generates a quick response (a control opens a gate) or generates deeper processing (a truck QR code triggers a lookup of load documents, which in turn adjusts warehouse inventory levels).

There seem to be three levels of processing depth that are significant. The first, “local processing”, implies control-loop processes that have action/reaction behavior and no enduring significance. Then we have “journaled processing” where events are still action/reaction but now must be recorded for review and analysis, and finally “transactionalized processing” where the event signals business changes that have to be reflected in other applications.

Local processing, combined with a local edge hosting model, means that the applications don't really have to conform to any particular architecture, and in fact are likely to be related to the devices involved in creating and acting on the events. Special tools and middleware are less likely to be required.

Where local processing involves either metro-centric or distributed-edge hosting, the applications would have to conform to edge computing development practices, set by the edge hosting provider. If event densities are low, some form of serverless hosting would be appropriate if the cost model worked out, and as densities rise, a container model is likely better.

With journaled processing, enterprises say that it would rarely be desirable to journal events locally except on a temporary basis while sending them to a deeper facility. They also say that journaled events would be collected either in the data center (most likely) or the cloud, and that they could be bulk transported rather than sent interactively. Thus, this model doesn't introduce a lot of additional architectural requirements or generate a need for specific tools/middleware.

Transactionalized processing is the complicated stuff, because there are a number of very different drivers. The “processing” may involve correlation of events in time and space, linkage to deeper databases and applications, and even extension of the control loop. It may require linking to the cloud, to the data center, to the latter through the former, or to both. Most potential event-driven applications likely fall into this category, though many of the current ones still represent the earlier and easier models.

It's also transactionalized event processing that introduces the other topics we'll cover in my series on edge computing. I'll summarize them here, and take each of them up later on.

The first of the topics is **edge portability and hybridization**. In transactionalized applications, the edge can be considered an extension of something else, so the obvious question is just how much software architecture at the edge has to draw on the architecture of what it extends. The biggest part of this question is whether we should look at the edge as a service as being an extension of public cloud software models.

The second topic is **dynamic edge distributability**. In some transactionalized applications, it may be necessary to dynamically distribute components among edge hosting points in response to changes in requirements or movement, addition, or deletion of event sources. This doesn't include backup functions, but rather changes in the nature of the event mission created by a changing set of event sources. Think gaming, where players might move their avatars into proximity and thus indicate a

common hosting point should accommodate the concentration.

The final topic is **edge security**. Not all edge models have the same security issues, and not all edge security issues will be obvious, particularly in complex distributed-edge situations. The emergence of edge computing as a technology, and IoT and the real-world synchronicity that it brings, could generate massive problems if we're not careful, and even perhaps if we are.

These are the topics I'm going to address in the rest of this series, but I'll do so by talking about a set of closely related edge-hosting models and relating the issues to each. The first of the series will be about **the edge as an extension of the cloud**, and I hope you'll be interested in the series and will comment freely on LinkedIn. I'll post the entire series as a CIMI Corporation white paper, available for download (at no cost, obviously) when the series is complete.

The Edge and the Public Cloud

Is “the edge” an extension of the public cloud? That may well be the biggest question we face in edge computing, because it determines what players are likely to dominate the evolution of edge computing, and who will frame how edge applications are written. Those factors may then determine just how fast we get a viable edge architecture, or whether we ever get one.

Edge computing and edge-as-a-service are related but not identical concepts, just as computing and cloud computing are. There has to be an architectural model for both to guide development, and the model can be obtained by extending the cloud to the premises using cloud-provider tools, or by taking a premises-compatible software suite and building applications that then run as containers or VMs in the cloud. In the former model, cloud providers win, and in the latter case the software providers win.

The winner of a race is the player that gets there first, obviously. Public cloud providers not only think that the edge is a part of the cloud, they're the only class of supplier in the market that's actually proactively developing the opportunity. Yes, there are others who are talking about the edge, but they're really just preparing to take orders, not to **sell** in a proactive sense. Not so the cloud providers; they're doing everything they can to own the opportunity.

The basic thesis of the public cloud providers' edge strategy is the classic camel's nose, based on the logical assumption that true edge applications don't exist yet, so whatever builds them will influence how the edge evolves. They are offering a way of hosting their cloud tools on premises equipment, the obvious effect of which would be to facilitate the growth of edge applications that would rely on their cloud APIs. That, in turn, would facilitate their offering actual edge-as-a-service down the line. Whether this would work depends largely on what's driving a particular edge opportunity, and what competing approaches are known to buyers.

There are two situations where the edge opportunity drivers would lead to a symbiosis between edge and cloud. First, where the buyer already has a strong commitment to the public cloud, which for enterprises likely means that they've created cloud front-ends to modernize legacy business applications. Second, where the edge mission lends itself to a combination of edge and cloud processing. It is likely that both these would have to be present to some extent at least, to promote the cloud providers' interest in the edge, particularly given that there are no true “edge-as-a-service” offerings, so local-edge, customer-owned, technology will have to do the hosting.

There are obviously a lot of enterprises with public cloud commitments, but what exactly are they

committed to? Today's public cloud applications aren't particularly event-oriented, and if events are the real future of the edge, then what's already running in the cloud isn't especially applicable to the emerging edge missions. Would enterprises adopt cloud tools they'd never used before, in local-edge applications they develop?

The “competing approaches” counterpoint's credibility depends on the same question, of course. If enterprises have to develop their own local-edge software, they could in theory use software tools not specific to the cloud. After all, they're developing local applications. In these cases, edge strategies would drift away from the cloud model, and we'd have a greater chance of what my opening blog in this series called the “local edge”, without any specific link to cloud provider tools.

The greatest mission driver for a cloud-centric edge is where event sources are highly distributed and/or highly mobile. This would make it difficult to justify premises edge hosting at all, and it's also true that some of these distributed/mobile edge applications aren't so latency sensitive as to require “edge” at all; the events could be fielded by the cloud. I worked with a transportation-industry player who had specialized cellular connection to its carriers, to report conditions like out-of-range temperature, impacts, or openings. These don't require low latency; they're alert rather than control applications. However, they also had some applications that involved traditional control loops, so their processing mixed cloud and edge-as-a-service requirements.

The cloud also has great potential for event-based applications that require coordination of events from different sources, different locations, or both. Where event sources are highly distributed and where process elements are therefore distributable, the cloud is the logical place to connect the edge dots, however edge hosting is accomplished. That would at least open the prospect of using a common platform for development of cloud and edge applications.

One area that could drive a new level of symbiosis between cloud and edge is artificial intelligence and machine learning. I noted in my first edge blog in this series that most transformational edge applications were likely to involve creating a synchronization between the real world and a “digital twin” created by an application. AI/ML could be extraordinarily valuable not only in creating that digital twin, but in creating insights from it. Even where the event sources are fairly concentrated, cloud AI/ML tools are more accessible than premises-hosted tools, according to enterprises, and the use of cloud tools could permit AI/ML elements to migrate edge-ward if edge-as-a-service is available or if cloud provider tools were extended to the premises via the cloud providers' private edge initiatives.

All this is good for public cloud providers, of course, and it ties in with their partnerships with the telcos, too. One thing that the cloud providers realize is that their efforts to create cloud/edge symbiosis will likely accelerate interest in edge-as-a-service, and that could create a risk for them if they're not prepared to exploit the interest. Rather than buy real estate in every “significant” metro area (the number of these varies depending on who estimates, but my figure is 3,100 for the US), why not partner with somebody who has the real estate to start with, meaning telcos?

The problem with this approach is that these partnerships are driven, from the telco perspective, by interest in having cloud providers host 5G elements. The requirements for this have little or no relationship to the requirements for generalized edge computing applications, which means that there is little chance that middleware tools developed for the 5G mission will have any value down the road, and that raises the obvious question of where tools for those general edge missions will come from. Are they already part of the cloud?

No, at least not IMHO, they are not. It's hard to prove that without an example, so let me propose the

following. If edge computing is driven by events, and if the largest number of event-based applications involve the creation of a digital twin of elements of the real world, then digital twinning is the primary feature required for whatever edge applications emerge. Cloud providers don't offer anything in particular in that area today, and that is their greatest risk.

Digital twinning would use events to feed into an AI/ML framework that would then create a kind of lightweight abstraction representing the real-world target. That framework would then be used to draw insights about the current and future behavior of the target, and perhaps to invoke control responses to change that behavior. It's not an application as much as a middleware suite that would then be used to build applications. Think of it as a modeling process.

The fact that cloud providers don't offer specific tools to facilitate edge-enabling applications doesn't mean that cloud and edge aren't intertwined, only that the relationship isn't optimized today. That, in turn, means that others could seize control of the edge by creating the tools that **would** facilitate those edge-enabling applications, like my digital twinning example. The question is whether any others will.

It seems pretty clear that the network operators aren't going to transform edge-think. Some, like AT&T, seem to have at least some handle on the potential of edge computing, and even on how the edge dynamic could develop. However, even AT&T is looking at alliances with cloud providers to realize an edge vision, and some of their deals have ceded AT&T projects and staff to those cloud providers. That suggests they don't intend to push things on their own, and the reason is probably the operators' lack of cloud expertise. Things like NFV, the operators' vision for software hosting of network features, don't have any real role in edge computing.

IT vendors could be another source of edge power. Dell, HPE, IBM/Red Hat, and VMware are all potential giants in the edge space because it **is** likely that early edge deployments will dominantly be on premises rather than as-a-service. An edge model designed for premises hosting could be easily translated to a cloud deployment on containers or VMs, and that means that these vendors would be the only likely source of an edge platform architecture that didn't mirror some public cloud provider's tools.

The deciding point here, I think, is 5G. 5G is the only “application” of edge computing that is clearly funded and deploying, in the form of edge hosting of 5G elements. The network operators' relationship to the public cloud providers almost guarantees that 5G will be deployed on the public cloud, and that may be due in large part to the fact that those IT vendors I've cited haven't created a compelling 5G story.

Since they've missed 5G already, the IT vendors' only shot at edge dominance is IoT-related missions like digital twinning, which those vendors have also so far ignored. Unless that changes during 2021, I think attempts to wrestle the edge architecture from the public cloud are unlikely to succeed, and we can say that the edge of the future is an extension of the cloud of the present, but a future extension that's not yet been realized or even suggested by the cloud providers themselves.

The Edge Software and Middleware Model

Wherever “the edge” is hosted, what gets hosted is software. If there is such a thing as “edge software” as a subset of “software” or “cloud software” then there has to be development, and development has to be based on some set of what's popularly called “middleware”, a toolkit that facilitates development by providing a means of implementing important tasks likely to be a part of most applications. What are

the factors that determine what this middleware toolkit contains, and do we have examples of any of it? Those are the questions we'll address in this third in our edge computing series.

I proposed in earlier blogs in this series that edge computing was justified by the requirement for low latency in processing. I also proposed that the low-latency missions were almost surely **event-driven**, because the primary driver of latency sensitivity is the need to couple processes to real-world conditions that won't wait for you to get around to them. We have event processing software models today, so let's start by looking at them to see what works best.

Enterprises and even vendors today tend to associate event processing with public cloud functional/serverless computing, but those are public cloud features and rather than supporting edge computing's latency requirements, they actually add to latency. A serverless element is loaded on demand, and the load latency usually exceeds the latency improvement that could be had by moving an application closer to the edge source. There are multiple cloud providers offering serverless, and multiple service options, but latencies can range from over 100 milliseconds to nearly a second when a function is "cold started". In addition, serverless pricing favors applications with modest event generation rates, both because higher rates of event generation would suggest that the process elements should stay resident and because pricing models for serverless work only with fairly low event duty cycles.

Container hosting of event processing, which keeps the process elements loaded and ready, adds relatively little (perhaps 5-10 milliseconds) to overall latency. If event duty cycles are high enough, containers would also likely be cheaper than serverless, but the big advantage of containers here is that the linking of events to containerized processes doesn't require cloud provider middleware, so it's possible to use generalized event software in the cloud, on premises, or in hybrid form without paying for special cloud event processing or event routing tools.

Cloud providers do offer event processing, but you won't find it in the list of products on their websites. Instead, they offer IoT support, which validates my view that IoT and some model of digital twinning is the key to the edge. However, the specific IoT linkage to event processing means that there isn't a clear connection between event-oriented service activities of the type that 5G would generate, and public cloud middleware tools. It also suggests that, at least for now, cloud providers aren't looking as much toward generalized edge computing as perhaps they could or should be. Or maybe they're looking at it through the wrong lens.

There's nothing generally wrong with having an IoT bias for your edge strategy, given that IoT is a very likely application driver. The specifics are the issue. IoT's overwhelmingly about some sort of process or activity control today, and that particular mission is much more likely to stay on premises unless the event sources are highly distributed, which usually is not the case. Thus, we're not building toward a new and larger set of edge applications. Another issue is that much of the IoT support offered by cloud providers has more to do with the mechanics of the IoT environment, registration of devices, and so forth, than with the deeper requirements of applications.

What would differentiate "generalized edge computing" from cloud computing or event processing? It would have to go back to the points of my first edge blog. Component/application portability and hybridization, dynamic distributability, and security are all things different for the edge than for the cloud. Once these elements were sorted out generally, there would also be specific things relating to the mission requirements of the primary drivers. For example, if we accepted my suggested digital twin approach, we could expect to see tools to establish and sustain the twin, and to use AI/ML to draw inferences from it and feed back those inferences in the form of real-world changes via "actuators".

Concentrated event sources, like warehouses, factories, or hospitals, would be candidates for implementation using traditional tools and practices. Even event-processing tools like those for complex event processing, or CEP (including open source tools like Esper or Open CEP) or streaming tools (like Tibco Streaming or Fluvio), would serve many of these applications. However, if the concentration of event sources happens to concentrate multiple systems of event sources, or if the sources are distributed or mobile, then it's likely essential to create a model of the real world that events from multiple sources update as needed. This model, as I've previously mentioned, is then the basis for both controlling things and drawing insights.

The “real world” is a distributed, interreactive, system. Most event processing software deals with a single stream of events, though some CEP software could be adapted to the mission if the real-world system wasn't too complex. As the complexity builds and the pace of event generation expands, it becomes more complex to handle with any of the current software.

The scope of a digital twin, or of any real-world-related application that's a candidate for edge hosting, is the scope of the real-world system that it represents and controls. To the extent that the application's real world is distributed, it's likely that the application will also have processing depth, meaning that an event flow might pass through multiple processes designed to deal with more stringent latency requirements near the edge and more forgiving requirements deeper. It's also likely that the application will look like a series of interconnected application components arranged in a structure of some sort.

In distributed-event-source or mobile-event-source applications, the obvious problem is process coordination. If the event source is concentrated and static, conventional event-flow strategies will work fine, even if the process depth spreads components from edge to cloud to data center. Where there's a distributed or mobile source, the latency requirements associated with each step through the process depth required map to different hosting places depending on the source of a specific event. Each process step is essentially an autonomous element that has to respond (perhaps) to the event and also coordinate (perhaps) with other steps in the process and even steps in the processing of other events.

Dare I point out that this isn't far from the service model that the TMF's NGOSS Contract described and that my ExperiaSphere project expanded upon? Each event might first intersect an edge process, which might be local to the source or edge-as-a-service-hosted, and there generate any time-critical response, then dive deeper in the form of either a repeat of the original event sent to a deeper process, or in the form of a newly generated event with a deeper destination. Applications would then be a series of state/event systems that defined their own process-to-state-event relationships in a table.

A good candidate for generalized tool for edge/event processing would be a framework to establish just that, and since there is a good deal of affinity between that and the telco service world, it is very likely that this tool would also be suitable for 5G and even other services. The real world is distributed and asynchronous, just like a network, and just like a network's lifecycle management needs to be orchestrated holistically, so does any real-world, digitial-twin-based, application of the edge. Thus, the edge-orchestration tool would not only support 5G, but it could be the basis for other real edge applications.

If this is true (which I obviously think it is), then we've missed an important piece of both IoT and edge computing by failing to address the question of how we model real-world systems and orchestrate components and their lifecycles within that model. That offers some aspiring edge giant to step up and provide that missing piece, and by doing so perhaps gain a major market advantage.

Edge Lessons from Existing Applications

What can we learn, if anything, from those applications already considered "edge computing"? We have cloud applications that are event-driven today, and some of them aren't even IoT. We also have some applications that might not seem event-driven at all. Obviously, we have local edge computing in place for most of the "IoT" applications enterprises run today. If the popular notion of edge computing, which is "edge as a service" is to be realized, we clearly have to go beyond that which is already done. That doesn't mean that the evolving edge model should ignore the opportunity to support and improve current applications.

Today's IoT applications fall into two categories, those that represent traditional-industry, fixed-facility, deployment of IoT elements, and those that represent either mobile IoT elements or IoT elements that represent high-value, low-density, objects. The former are almost always linked to local controllers and local edge hosting components that provide a very short control loop. These may then be linked back to cloud or data center applications that can journal events, process transactions, etc. The latter are now divided between "phone-home" missions that report condition exceptions or occur at regular intervals, contacting the data center, and cloud-connected event handling.

One thing that, according to users themselves, characterizes these current applications is that there is a limited event volume, either because events are naturally infrequent or because local processing is doing some form of summarization. It is true that some fixed-facility IoT applications will journal all events, but often this takes place locally and is transferred in batch form rather than in real time. Users will ensure that any real-world control generated by an event is quickly fed back, avoiding process paths that add a lot of latency. Thus, applications are designed to push any real-time processing closer to the event source.

Right now, these applications face what might be called a "latency dead zone". Local processing of events will produce a very low latency, but obviously don't use hosting in a cost-effective way. If you pass through local processing, the next available step (cloud or data center) has a significantly higher latency associated with it, so if it's necessary to push real-time processing closer to the event source as just suggested, then pushing it all the way to a local edge may be the only option. Edge-as-a-service could offer an intermediate hosting point and fill in the dead zone.

Stepping outside IoT, we have two broad application classes that are often paired with cloud-hosted event processing, and which aren't real-world/real-time related. One is log analysis and the other is video analysis, and they obviously represent a class of applications (the "analysis-source") that might also be the way that some local-edge IoT applications communicate inward toward core applications.

The need to keep control loops short is the primary driver of moving things toward the edge. When edge analysis is performed, it allows summary information to be taken deeper, insulated latency issues, the need to preserve relative timing, and potentially high volumes of events that could drive costs up. Applications like log and video analysis are among the major drivers of cloud event processing and serverless computing today, but these applications have been visible/available for years and have not revolutionized or galvanized edge computing. For that, I believe, you still would need IoT, but perhaps a different view of IoT than we have today.

The great majority of IoT applications today are highly confined, and that means that they are well-served by the "local edge", computing resources located on premises and close to the event sources.

The latency dead zone I referred to is closed off by the suitability of these local edge resources, for the simple reason that there's enough event generation in one place to justify that local processing. We could surmise (but I can't prove it) that applications that aren't well-suited for local edge processing, ones that need edge-as-a-service, can't develop today because the facilities aren't available. One potential proof point of that claim is the “smart city” concept.

A metro area is clearly a distributed real-world framework, but there are few examples of smart-city technology that even approach the digital-twin model. We have some examples of metro traffic control, metro power control, and so forth, but not of a single system that embraces all the real-world things we'd like to manage. In effect, we've substituted application-class segregation of event processing for facility segregation. That suggests that there is indeed a lack of an overall model for digital twinning.

What about serverless orchestration tools like Amazon's Step Functions? Microsoft and Google both have equivalent capabilities, so this represents the cloud provider way of handling the orchestration of event-driven tasks. The tool is best visualized as a graphic representation of just what Amazon's name for it suggests, the steps associated with some event process. You can define a linear stepwise sequence, branching, and so forth, and a step function can (as a step) activate another step function by generating a trigger event.

For digital twinning, it's the step-within-step capability that would be important. A complex system could be defined by having each of the simple pieces first defined as steps, and then combining/uniting the steps using trigger events. Or it could be in theory; it's not clear whether this would be practical because of a combination of difficulty in visualizing the whole picture, and difficulty in maintaining all the states of the individual steps and the collective system. This is pretty much the same issue that would impact CEP software.

Note that I didn't cite the serverless relationship to Step Functions (or its competitors) as a problem. That's because you can orchestrate containerized elements too. That opens the possibility that some local facility or event-source with edge processing could feed an event to the cloud to trigger correlation there. In other words, you could create individual step orchestrations on local edge processing, and let the complexity of merging them into the real-world system be handled in the cloud.

Step Functions, and other workflow orchestration tools, including Apache Airflow, have a common issue in the fact that, for digital twinning, they don't have a built-in mechanism for keeping track of trigger events that link the individual atomic systems/steps. If a given piece of the digital real-world twin has to generate an event to notify another part, whether higher-level or not, that piece is outside the scope of the systems. You can generate these events in your processing, but there's no easy way to track the way the whole structure of steps is linked, other than manual. That means the process will be difficult to scale up without major risk of errors.

We can do better, I think, by thinking of the entire digital twin as a graph of a set of finite-state machines (see this excellent article on using FSMs for event-driven applications) representing the systems that make up the real-world target. Traditional FSMs can't handle a complex real-world digital twin any better than a step-function-like system would, for the same reason. However, there's such a thing as a hierarchical state machine (HSM) that could fit the bill, and a few open-source projects support HSM in some form.

If you look at the HSM reference in the last paragraph (and you understand software), you'll likely see that while HSM would work to model a real-world system, the tool would have to take major steps to allow such a model to be created and maintained without extensive work and software knowledge. If

there's any such tool on the market today, I can't find it.

I have to admit that I still prefer the notion of adapting the TMF NGOSS Contract data-model-driven event-to-process mapping as the basis for creating digital twins. The model would represent the elements of the twin, and each element would have an independent state/event table that would decide how to process events and also when to signal an event to another element. The problem is that nobody really implemented this approach, and it postulates a hierarchical structure of elements rather than a generalized graph that would allow any element to signal any other via an event. The HSM approach would seem to address this, but with a level of complexity that I think would need to be resolved with a specific software package.

Where does this leave us? We have tools today that could be said to light the way toward a digital twinning approach to edge computing applications, presuming they were event-driven. Lighting the way doesn't get us there, though. As things stand, we are at risk for two bad outcomes. First, multiple edge application vendors could take totally different approaches, which risks the operational efficiency of the edge. Second, and worse, no suitable strategy might emerge at all. That could mean that it will be difficult for edge computing to meet our expectations.

Ecge Networking and Security

This is the fifth and final blog in my series on edge computing, and in it we'll talk about two critical issues. First, how do we network “the edge” to realize the capabilities buyers expect. Second, how do we secure the edge, given that its real-world relationship target makes it a profound security risk.

Any discussion of edge networking has to address the question of just where “the edge” is. Local-edge networking, the connection made to and from an on-premises processing point, is obviously the user's responsibility. However, the user has to ensure that this connectivity doesn't compromise the latency improvements that edge computing is aimed at delivering.

The great majority of connections **to** a local edge will be made using a local technology, like wires, WiFi, or an IoT protocol like ZigBee. From the local edge inward, connections could be made in a variety of ways, including the WAN options of the Internet (and SD-WAN) and the corporate VPN. These WAN options would be applicable to cloud processing, data center processing, and any edge-as-a-service being consumed. Whatever option was selected, it would be critical to maintain the performance of the connection, both in terms of capacity (to avoid congestion delay) and in terms of intrinsic latency, which is a combination of propagation delay and handling delay by devices. All of this is already understood by enterprises.

The big network question isn't related to local-edge at all, but to edge-as-a-service, and it's not so much how you connect to it, but how traffic is handled **within** the edge. Recall from past blogs in this series that “the edge” as a service is almost surely a metro-area service. That means that edge hosting resources would be distributed within a metro area in a variety of ways, likely determined by the total edge opportunity within a metro area. For example, it might start with a single metro process point, and expand as opportunity grows to include distributed hosting points closer to major traffic sources.

The key to metro edge success is the network, because it's critical that the metro edge hosting resource pool be fully equivalent, meaning that you could pick a hosting point from the pool at random and still fulfill the edge hosting SLA, including latency. That argues for a fabric-like connection model, with a

lot of capacity between hosting points and a minimum of handling variability—a mesh. You also need the metro fabric to couple to data center switching at each hosting point, and the same level of meshing at the switching level to avoid any variability of latency and QoE. The better this whole network fabric is, the bigger the metro edge can be without compromising the SLA, and the more efficient the pool of resources would be.

Edge computing is potentially the biggest driver of metro connectivity, and it's almost certain that it will contribute enough (via things like 5G feature/function hosting) to make the metro network the most important piece of the network overall. If cloud providers dominate edge-as-a-service, then much of the traffic from the metro areas will go to cloud provider data centers, and the larger metros at least will surely have direct fiber connectivity. I expect that more and more traffic, and particularly high-value traffic, will bypass any IP core networks completely, or will connect to the core only to reach Internet destinations.

CDNs have already shifted the balance of traffic away from the core network, and edge-as-a-service would make the shift decisive. If gaming or autonomous vehicles were to become major edge applications (which is not certain, despite the hype) then we would likely see metro-to-metro meshing as well to accommodate distribution and migration of edge application components. That would tap further traffic from the IP core. All of this indicates a focus on deployment of metro router/metro fabric complexes and a reduction in traditional core routing requirements.

Finally, it's possible that edge computing, if widely successful, could combine with cloud computing to change the whole notion of a VPN. If edge and cloud combine to become the connection path for both users and “things”, then the concept of a corporate VPN would reduce to be nothing more than connections to the Internet/edge and connections between data centers. Think about that one, and you can see that it would bring a truly seismic change in the industry!

That might happen through something that's already bringing about change, which is virtual networking. The connectivity dimension of edge computing, and the fact that it could change VPNs, mixes edge impact with things like SD-WAN. If the future of edge networking and cloud networking diminish traditional VPN roles, they don't diminish the need to manage connectivity better, and differently. Virtual networking can manage multi-tenancy; those needs arguably launched the whole trend. Virtual networking can quickly extend itself and contract itself to accommodate changes in application scope, too. And virtual networking can enhance security.

Edge impacts on security could surely be profound. Obviously, any complex distributed system has a larger attack surface, particularly when much of it is hosted on multi-tenant infrastructure. Network operators have shown little awareness of this in their NFV work, but fortunately public cloud providers have been focused on multi-tenant hosting security from the very first, and they've generally done a good job at the infrastructure level. The edge security problem is likely to arise higher up, in the middleware and application components.

Stringent latency constraints tend to discourage complex security tools, and if edge computing serves event-driven applications, the event sources themselves become attack points. In addition, the APIs that serve to connect application components from event source to ultimate processing destinations will have to be protected. All of this has to be done in a way that, as I've noted, doesn't compromise latency benefits, and it has to be consistent or there's a major risk of holes developing through which all manner of bad things can enter.

It seems certain that a lot of the protection here will have to come at the network level, something

public cloud providers already know from their multi-tenant work, work that relies on various forms of virtual networking. We should expect that edge applications would run in private address spaces, for example. Think of VPNs that seamlessly cross between premises local edge, edge-as-a-service, the cloud, and the data center. SD-WAN can offer that.

One significant security issue with edge computing, presuming proper tenant separation by address segmentation, is likely to be the introduction of malware components, a Solar-Winds-style hack. The risk is exacerbated by the possibility that many edge computing features will be based on middleware libraries that may or may not be fully secured. This risk would be mitigated if the primary cloud providers or major software vendors provided the edge middleware, but the sheer dynamism of the edge **might** make it difficult to spot offending elements because of shifting hosting points and traffic patterns.

Could zero-trust security, which is offered by a very few SD-WAN vendors, be a key to securing the connectivity of edge computing? I think it could, and that could be a major benefit, because whatever security can be offered at the connectivity level, whatever insights into bad behavior can be obtained there, will not only reduce risk but also reduce the need for security elsewhere, security that may prove difficult to provide without adding a bunch of new layers.

That introduces what I think is the biggest edge security issue, and that is complexity and its impact on operations. People are going to ignore this because many see edge-as-a-service as being nothing more than a kind of local cloud. The problem with that vision is that there's no justification for edge computing if it's simply relocated cloud computing. Latency-sensitive applications don't look like cloud applications; if they did they wouldn't be latency-sensitive. The applications, as I've noted in this series, will drive a whole new application architecture, an architecture that's highly dynamic not only within the edge provider's infrastructure, but in its hybrid relationship with users. Future edge applications, to justify the edge, will be more complex, and that's a problem.

The greatest potential application of artificial intelligence and machine learning may well be edge operations. We've had (including recently, with Akamai and Fastly) examples of how little issues can create global problems, and arguably these little issues come down to operator errors or process failures. In a low-latency world, the effects of blunders have low latencies too, and they multiply.

Internet security is almost as big an industry as networking, and that proves that if you don't build security into a technology architecture, you can't really hope to secure it. We will need to consider edge application security fully before we start deploying edge applications, period. Remember that edge computing is really about digital twinning and real-world application synchrony. We mess up the application and we mess up the real world. The consequences of security issues in edge computing are absolutely dire, and the risk of malware and hacking are perhaps even worse.

Let's close things out now for this blog series on the edge. Edge computing, if it exists as a differentiated strategy, has to serve a set of missions that aren't currently addressed. Those missions seem to focus on latency-sensitive IoT-like applications, particularly ones that require an application create and/or maintain a digital twin of a real-world system. There are two possible edge models, one where the edge is hosted by the user near the event source, and the other “edge-as-a-service” where it's hosted by a provider in a metro complex. Since we can already support the former, any transformation of computing that arises out of the edge would have to come from the latter.

Edge-as-a-service necessarily involves a set of tools, what we could call “middleware” or “web services”. These tools should provide a common model for building and running edge applications,

both local and as-a-service, and the model has to make security a built-in, not an add-on, property. Something like a Hierarchical State Machine (HSM) is the most promising of currently identified technology concepts to form the basis for digital-twinning applications.

A foundation of both the edge infrastructure and edge applications is a proper set of network tools to provide secure and agile connectivity among components, and in and out of the edge. If a lot of agility, tenant separation, and application security has to be provided at the edge, for applications whose structure and distribution will vary, then it seems likely that virtual networking will have to play a major role.

We already have SD-WAN tools that span from user to cloud to data center, and it would seem very logical to incorporate these tools in the edge, particularly if they can also provide zero-trust security. For IoT applications, of course, overhead and latency will be key issues since access bandwidth could be limited and it would make no sense to introduce latency through a virtual network when the applications' justification depends on low latency.

The final point about the edge is (OK, you probably guessed it!) we need a single edge architecture from the deployment of infrastructure upward to the deployment and operationalization of applications. The edge is going to be a mixture of applications, ranging from elements of telecom infrastructure for 5G through IoT, and on to gaming and more. If every application has its own toolkit, it's hard to see how we can avoid overspending on the edge, and operational and security challenges.

Some sort of “edge computing” is certain, because we already have it in IoT applications and more. Edge-as-a-service is likely certain too, largely because of 5G, but the kind of profoundly different edge computing we read about is far from certain. We need to have a viable edge model, and here we will almost surely end up relying on public cloud providers, unless software vendors act quickly to counter the cloud providers' early success. If we want the best of all possible edge computing model, we'll still have to work for it. I hope this series has, by laying out issues and options, improved our chances of success.