IT for CEOs & CFOs is published by House of Words Media Limited.

The current issue and full text archive of this journal is available on our main website at **www.creditcontrol.co.uk**.

Amadeus and Intent-Based Analytics

Sasha Ratkovic

Sasha Ratkovic
Chief Technology Officer and Founder
Asptra

Biography

Sasha Ratkovic is a thought leader in domain abstractions and intent-driven automation. As CTO at Apstra (www.apstra.com), he drives the architecture direction of the Apstra Operating System and is deeply involved in all aspects of the Apstra Product and Engineering efforts which enable network operators to realize log-scale improvements in CapEx, OpEx and capacity of their network infrastructure.

Sasha was the first to propose Group Based Policies at Openstack, a year before others adopted the approach and took leadership of the work group. Prior to Apstra, Sasha was a distinguished engineer at Juniper Networks where he led automation efforts for data centre products. Sasha holds a Ph.D. in Electrical Engineering from UCLA.

Keywords Intent Based Networking, Digital transformation, Apstra, Non-proprietary
Paper type Opinion

Abstract

In one of my favourite movies of all time, "Amadeus", there is a scene about the conversation between Mozart and the Holy Roman Emperor, Joseph II, which took place just after the first performance of the opera "The Marriage Of Figaro". During the course of conversation the Emperor "explained" to Mozart, in a fairly condescending way, that while the opera was excellent, it had "too many notes". He went on to suggest "Just cut a few and it will be perfect". Shocked, Mozart calmly replied "Which few did you have in mind your majesty?". A pregnant pause ensues, and the Emperor is only saved by a distraction which allows the change of subject.

This "change request" while it appeared on the surface trivial to the Emperor, turned out to be quite a challenge as it required the complete knowledge of the score (state/intent) and required composition skills (design/action). So why are we talking about this in ***IT for CEOs & CFOs****?*

Introduction

The main challenge facing people running compute/network/storage infrastructure today is composition. By composition we mean the ability to create a coherent whole, which is bigger than just the sum of its functioning components, and to sustain that coherency over time, in the presence of inevitable changes (Buddhism has figured this out a long time ago).

Smart composition (for instance, accounting for failures) can make dealing with failures easier, which is very attractive. Smart composition also allows for delivery of services without relying on proprietary hardware stacks.

IT for CEOs & CFOs is published by House of Words Media Limited.

The current issue and full text archive of this journal is available on our main website at **www.creditcontrol.co.uk**.

Modern data centres are scale-out computers. They need an operating system. They need Amadeus. This data centre operating system provides functionality for the data centre that is analogous to what a host operating system provides on a single machine today: namely, resource management and process isolation. Compute virtualization already does this for a single compute. But for data-centre-as-a-scale-out-compute you as a data centre operator first need to compose it and only then (and armed with that knowledge) you can provide resource partitioning. Even "cut a few notes" (or add few virtual networks and more capacity) while your first violins are on a sick leave (or ToR switch is rebooting for no apparent reason). So, what are some of the capabilities this system has to provide that are required to deal with composition @scale?

The traditional approach to deal with scale is decomposition. The "everyone knows everything" approach doesn't scale. You have to distribute the knowledge about the desired state and let each element determine how to reach that state, so you avoid having a need for centralized decision making. In the context of a data centre network components are given declarative specification along the lines below:

> *"You are a spine switch, these are your resources, these are the facts about your neighbours and environment you need to know about, so now be one and act like one. Let us know how you are doing, and you will be informed if any of the facts change."*

Need for dealing with change

The last sentence implies the need to deal with change. This is the most fundamental requirement. Everything needs to be treated the same way, as a change. Even initialization.

Especially initialization. In the absence of it your clean greenfield is going to turn into brownfield overnight, with the first change request.

Change coming from the operator

Where does this change come from? It can come from the top, from the operator (intent), in the form of a new business rule. This change can be related to any aspect of service lifecycle:

1. **Design** – I need to modify the design to support 30% more servers and reduce oversubscription to 1:1. Or I am starting from scratch and need to connect 5000 servers with 2:1 oversubscription.

2. **Build** – I would like to use an IP un-numbered addressing scheme for my fabric links and use alternative vendor devices in the design for incremental capacity, and I want to decommission some devices that were giving me grief in the past. I want to add new L3 security zone connecting a number of L2 domains.

3. **Deploy** – I want to deploy changes incrementally, during a maintenance window.

IT for CEOs & CFOs is published by House of Words Media Limited.

The current issue and full text archive of this journal is available on our main website at **www.creditcontrol.co.uk**.

The spine switch from the previous paragraph subscribes to a place where it can learn about the changes it cares about such as the new neighbours or the link IP numbering scheme being changed, etc.

If your network is relatively small, you treat your switches as pets (and have a name for each of them) and if the rate of change is low, you can do this in your head. However, if you need to do this at scale, where you need to track thousands of ephemeral ghost-like constructs in real time, you need software to do it for you, and this requires the ability to programmatically reason about the intent and state of the infrastructure. You still retain control over red and green action buttons, but analysis of what is going on is automated. To do such analysis in real time, a supporting system must be able to query the source of truth on a continuous basis, and be notified of any changes of interest.

Reasoning as applied to examples (1 -3) above means:

- Given my reference design, how do I add more capacity? Assuming you were using L3 that means adding new leaf switches, subject to port count constraints, otherwise adding another tier and moving to a 5 stage clos.

- How to reduce oversubscription? Add more leaf spine cables, or add more spines, subject to constraints.

- Manage resource isolation/reuse according to intent while creating new security zones: allocate required resources to new security zones (VRFs, IPs, VNIs, VLANs …), enforce uniqueness where required (for example, VNIs), or re-use as appropriate (for example, IPs).

- Ensure a new vendor's device has capabilities to play a role in the reference design.

- Generate new configurations for affected devices to reflect new topology.

Change coming from the infrastructure

Change can also come from the infrastructure. For example, you may observe a symptom that a device appears to have failed. But you need to reason about this carefully. Has the device failed, or has the agent that implements the liveness check failed? Or did you simply lose the connection to it? Or is there even a false positive? To find the root cause, you need to reason about composition (the fact that there exists a coordinated effort on the part of the device, liveness agent, connectivity and their inter-dependence) and individual statuses. There is no self-healing in the absence of the knowledge of intent. This will also help deal with <link to> grey failures which are notoriously difficult to identify.

As far as raw telemetry goes, you don't start with what is easy to measure. Carefully select aggregations in order not to obscure important details. On the other hand, capture too many details and you cannot reason about the data – and aggregation about what is important has two dimensions. One is time. Do I care about instantaneous state, or how it evolved over time and over which time period?

How long was it above some threshold? What was the variance? The second is space. I may care about only fabric links, in a specific pod, attached to leaves running a specific version of software (known to have issues) and supporting important customers. This enables you to observe complex temporal and spatial signatures with increasingly greater semantic content.

Now to get there you need to be able to declaratively tell the system how to extract the knowledge out of raw data. Writing imperative big-data programs to achieve this doesn't scale in the presence of change. You need to recognize symptoms as complex signatures, and root cause them. All of this requires programmatic reasoning about the intent as a single source of truth.

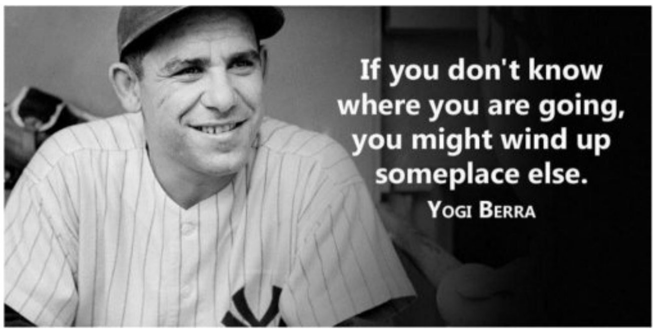
To summarize, intent serves as an explicit behavioural contract between the business rules and the operational state of your infrastructure, that is continuously enforced in the presence of change. Change in the business rule may require change in the infrastructure and you need an easy way to query infra to get what needs to be done. Change in the operational status of the infrastructure needs reasoning about the business rules still being honoured.

Fear of change

Ironically, one of the obstacles to accepting the need for dealing with change is resistance to change. Intent Based Networking has attracted a lot of attention recently. Chuck Robbins, the CEO of Cisco stood up on stage and declared that "Intent Based Networking will redefine networking for the next 30 years". The interest from customers has been overwhelmingly positive. Invariably, some of this attention comes in the form of intent washing, followed by intent bashing. This kind of attention, ironically, comes as a result of lack of attention to detail, and lack of the willingness to surrender pre-conceptions, which hinders progress.

In these oversimplifications intent is simply equated to "declarative" specification from which configuration is derived. If that were the case I would be sceptical as well. There is more to IBN than just that, as defined by analysts, and as exercised by actual practitioners and people working on it. The most important piece is actually validating that a declarative specification actually delivered what was promised (behavioural contract described above)

It may be worthwhile to look for a kernel of truth underneath the hype. It starts with the realization that imperative approaches do not work at scale. Period. An imperative approach executes steps that get you somewhere, but it doesn't allow you to define "somewhere".


If you don't know where you are going, you might wind up someplace else.
Yogi Berra

As Yogi Berra once said: "If you don't know where you are going you might wind up someplace else."

It starts with understanding the required supporting technology to enable what intent-based networking promises. Forget about terminology ("intent") and look for a set of capabilities required to deal with composition @scale such as the ability to programmatically reason about single source of truth in the presence of change which enables all the goodness described in this article. If these elements are not there and someone utters the word "integration" (or hides multiple sources of truth behind smoke and mirrors), I would recommend you run away as fast as you can.

But technology alone is not a sufficient catalyst for this change. It is our collective response to opportunities presented by technology that will drive this change. In addition to new tools there is also the opportunity to evolve your skill-set. Five years from now, new technologies and new capabilities will show up, and your best reference design and much of your hardware will become obsolete. But that is ok if we invest in our composition skills along with the tools to support it. Future system's capabilities may change but the nature of the composition problem does not. Just ask Amadeus. You will acquire the agility to develop your own designs leveraging features supported by devices from established vendors and open alternatives.

In conclusion

Intent as an abstraction by itself does not reduce overall system complexity but it is a critical piece as it reduces the amount of detail you need to deal with at one time. As a result, as composer @scale you get to pay attention to details even in what is essentially the big picture.

Fear of change is human. As a young boy character in Game Of Thrones asked his father: 'Can a man still be brave if he's afraid?' 'That is the only time a man can be brave,' his father told him. Change is not our enemy, it is a fact of life. A future in which a modern day Amadeus cannot react to change is our enemy.