Engineering Simplicity

Whitepaper

JUNIPER APSTRA-ARCHITEKTUR

Bewältigung schwieriger Probleme im Datencenter mit speziell entwickelter Automatisierung für den gesamten Netzwerklebenszyklus

INHALTSVERZEICHNIS

KURZFASSUNG

Der Aufbau, die Bereitstellung, der Betrieb, die Verwaltung und die Fehlerbehebung eines Datencenter-Netzwerks können kompliziert, teuer und ressourcenintensiv sein. Laut einem Gartner-Bericht aus dem Jahr 2019 ist die Simplifizierung der IT-Infrastruktur heute die wichtigste strategische Priorität für Unternehmen. Juniper® Apstra wurde speziell für die Automatisierung des gesamten Lebenszyklus eines Datencenter-Netzwerks entwickelt.

Dieses Dokument beschreibt einige der schwierigsten Probleme, mit denen Netzwerkteams in Datencentern heute konfrontiert sind, und erklärt, wie die Apstra-Architektur jedes dieser Probleme löst.

Einführung

Die Architektur- und Betriebsteams von Datencenter-Netzwerken stehen heute vor vier großen Herausforderungen:

- **Zusammenstellung**. Es kann äußerst schwierig sein, ein kohärentes und reibungslos funktionierendes System aus Infrastrukturkomponenten verschiedener Anbieter mit jeweils unterschiedlichen Fähigkeiten zusammenzustellen. Juniper® Apstra schafft ein kohärentes Ganzes – eine Blaupause –, die alle Informationen enthält, die für den Einsatz des Systems auf der Grundlage der erklärten Ziele erforderlich sind. Die Blaupause wird in die physische Infrastruktur übertragen und ermöglicht es dem Betreiber, zuverlässige und benutzerfreundliche Services bereitzustellen.
- **Verlässliche Veränderungen**. Veränderungen sind eine Konstante im Datencenter. Sie können von einem Betreiber, der versucht, neue Services hinzuzufügen oder die Kapazität zu erweitern, bedingt sein oder durch die Infrastruktur, in Form von Störungen. In beiden Fällen müssen Betreiber einen besseren Weg finden, mit der Veränderung umzugehen. Laut einer Reihe von Studien aus den letzten Jahren[1] sind 65 % bis 70 % der Netzwerkausfälle auf menschliche Konfigurationsfehler bei der Durchführung einer geplanten Änderung zurückzuführen.
- **Erweiterbarkeit**. Um Innovationen zu fördern und dem unumgänglichen schnellen technologischen Wandel Rechnung zu tragen, müssen Datencenter in der Lage sein, neue Merkmale und Funktionen einfach und reibungslos einzuführen. Im Zuge der technologischen Entwicklung bringen die Anbieter Innovationen und neue Funktionen auf den Markt, die die Betreiber nutzen, um wettbewerbsfähig zu bleiben. Diese Entwicklung führt zu einer Änderung der Entwicklungszeit, sodass neue Verhaltensvereinbarungen – oder Referenzdesigns – für die zu implementierenden Services erforderlich sind. Mit anderen Worten: Ihr Strukturgefüge muss erweiterbar sein, um weitere Innovationen zu unterstützen oder neue Anforderungen zu erfüllen.
- **Skalierbarkeit**. Um all dies in großem Umfang zu bewältigen, müssen Datencenter-Netzwerke in der Lage sein, zu wachsen und Innovationen und zunehmender Komplexität gerecht zu werden.

Das primäre Ziel der Apstra-Architektur ist es, diese Probleme direkt anzugehen.

Die zentrale Herausforderung: Strukturgefüge

Die größte Herausforderung für Betreiber von Rechen-/Netzwerk-/Speicherinfrastrukturen ist heute das Strukturgefüge. Ein Strukturgefüge bezeichnet die Schaffung eines kohärenten Ganzen, das größer ist als die Summe seiner funktionierenden Komponenten, und die Aufrechterhaltung dieser Kohärenz im Laufe der Zeit in Anbetracht unumgänglicher Veränderungen – und das alles bei der Bereitstellung von Netzwerkservices für die Verbraucher. In seinem Kern ist Apstra so konzipiert, dass es die Herausforderungen in Bezug auf das Strukturgefüge meistert.

Moderne Datencenter sind Scale-out-Computer. Sie benötigen ein Betriebssystem, das ähnliche Funktionen bietet, wie sie heute ein Host-Betriebssystem auf einem einzelnen Rechner bietet: Ressourcenmanagement und Prozessisolierung.

Die Computing-Virtualisierung ermöglicht dies bereits für einen Single-Server-Rechner. Aber für ein Datencenter als Scale-out-Computer muss ein Datencenter-Betreiber dies erst zusammenstellen; erst dann können Sie eine Ressourcenpartitionierung anbieten.

[1] Dazu gehören unter anderem: https://www.computerweekly.com/news/2240179651/Human-error-most-likely-cause-of-datacentre-downtime-finds-study; https://www.networkworld.com/article/3142838/top-reasons-for-network-downtime.html; https://www.ponemon.org/library/national-survey-on-data-center-outages.

Warum ist ein Strukturgefüge so problematisch?

Es gibt zwei Hauptgründe, warum Strukturgefüge so große Herausforderungen sind. Zum einen können die Elemente, aus denen sich ein kohärentes Ganzes zusammensetzt, von verschiedenen Anbietern stammen, unterschiedliche Fähigkeiten haben oder nur über verschiedene APIs verfügbar sein.

Zum anderen müssen Sie Ihre Infrastruktur möglicherweise so zusammenstellen, dass sie eine Vielzahl von Services bereitstellt, die jeweils mehrere funktionale Aspekte wie Erreichbarkeit, Sicherheit, Benutzerfreundlichkeit und Verfügbarkeit aufweisen.

Wenn Sie mehrere Serviceinstanzen einrichten, können Interaktionen zwischen diesen Komponenten zu Serviceausfällen führen – es sei denn, Sie wissen genau, wie Sie diese Services in Durchsetzungsmechanismen abbilden können. Da sich die Fähigkeiten Ihrer Infrastruktur im Laufe der Zeit weiterentwickeln, müssen Sie in der Lage sein, Innovationen zu nutzen und einzuführen, ohne die bestehenden Systeme zu beeinträchtigen.

Apstra meistert diese Herausforderung mithilfe von Referenzdesigns

Das Schlüsselkonzept, mit dem die Apstra-Architektur den Herausforderungen von Gefügestrukturen begegnet, ist das Referenzdesign. Ein Referenzdesign ist ein Verhaltensstandard, der definiert, wie der Geschäftsziele, der Intent, des Benutzers auf Durchsetzungsmechanismen abgebildet werden und welche Erwartungen erfüllt werden müssen, damit das Ziel als erreicht gilt.

Die Referenzdesigns regeln:

- die Rollen und Verantwortlichkeiten der physischen und logischen Komponenten,
- die Zuordnung von Services zu Durchsetzungsmechanismen,
- die Erwartungen, die erfüllt werden müssen (d. h. zu beobachtende Situationen).

In einem Referenzdesign sind die Rollen und Verantwortlichkeiten der physischen und logischen Elemente genau definiert, was wiederum den Umfang der Modellierung einschränkt und die Spezifikation eines minimalen, aber vollständigen Modells ermöglicht. Ein Referenzdesign bestimmt auch, wie der Intent in Durchsetzungsmechanismen abgebildet wird. Das Verständnis dieser Zuordnung ermöglicht die Automatisierung der Fehlerbehebung und bietet leistungsstarke Analysen, die auf dem Wissen über die Zusammenstellung des Systems beruhen, im Gegensatz zum Reverse Engineering der Vorgänge in Ihrem Netzwerk.

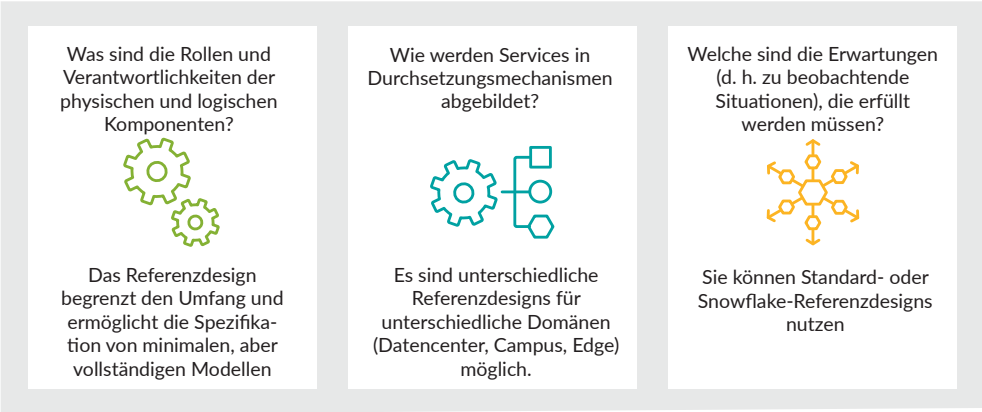

Abbildung 1: Das Referenzdesign ist die Grundlage für die Verhaltensregeln.

In verschiedenen Referenzdesigns kann derselbe Intent mit anderen, möglicherweise neueren und innovativeren Mechanismen durchgesetzt werden. Das Verständnis dieser Zuordnung ermöglicht eine Ursachenermittlung, die den Betreibern einen direkten Hinweis auf die Ursache von Problemen gibt, die sich auf den Service auswirken.

Das Referenzdesign spezifiziert auch die Erwartungen, die zu überprüfen sind. Ein Referenzdesign kann z. B. vorsehen, dass jedes Leaf eine BGP-Sitzung mit jedem Spine haben sollte. Auf diese Weise können fehlende und unerwartete BGP-Sitzungen leicht identifiziert werden.

Die Definition eines Referenzdesigns ist eine wertschöpfende Tätigkeit, die von einem Netzwerkexperten durchgeführt werden muss. Bei Apstra ist das Fachwissen ein expliziter Teil des Systems und nicht etwas, das nur im Kopf eines Experten existiert. Dies ermöglicht es, Fachwissen zu gestalten und in das System zu integrieren, anstatt es fest zu codieren.

Wissen ist Macht: Verlässlicher Umgang mit Veränderungen

> *„Es ist nicht die stärkste Spezies, die überlebt, und auch nicht die intelligenteste, sondern eher diejenige, die am ehesten bereit ist, sich zu verändern.“*
>
> – Charles Darwin, 1809

Veränderungen sind für Datencenter unvermeidlich, daher ist ein zuverlässiger Umgang mit ihnen immer von größter Bedeutung. Veränderungen haben zwei Ursachen:

1. **Die Ziele des Betreibers**. Ein Betreiber möchte vielleicht einen neuen Service, z. B. ein virtuelles Netzwerk, einrichten, eine Ressource zur Infrastruktur hinzufügen/daraus entfernen oder eine Änderung an einigen Richtlinien vornehmen. Der Betreiber hat die Kontrolle über eine Änderung, sobald diese eingeleitet wird. Apstra verwendet eine zustandsabhängige Orchestrierung, um die Absichten eines Betreibers zuverlässig umzusetzen.
2. **Ausfälle in der verwalteten Infrastruktur.** Änderungen können auch von der verwalteten Infrastruktur in Form von Ausfällen (z. B. übermäßige Paketverluste oder Ungleichgewichte im Datenverkehr) ausgehen. Ein Betreiber hat keine Kontrolle über diese Veränderungen, und die Menge der Rahmenbedingungen ist oft überwältigend. Apstra bietet Intent-based-Analytics, um Betreiber bei Änderungen des Betriebszustands zu unterstützen.

Die zuverlässige Bewältigung beider Arten von Änderungen hängt davon ab, dass Sie den Zustand Ihrer Infrastruktur kennen. Zwei Bedingungen müssen erfüllt sein:

1. Wissen muss im Kontext interpretiert werden, was bedeutet, dass die Beziehung zwischen Bedingungen und Erwartungen erkannt wird, wie im Abschnitt Kontextmodell beschrieben.
2. Das Wissen muss zeitgemäß sein, d. h. es muss die aktuellen Bedingungen widerspiegeln, wie im Abschnitt Überwachung und Benachrichtigung in Echtzeit beschrieben.

Statusbasierte Orchestrierung

Die statusbasierte Orchestrierung sorgt dafür, dass Änderungen, die vom Betreiber ausgehen, mithilfe von Intents zuverlässig durchgeführt werden. Ein zustandsbasierter Orchestrierungsablauf ist in Abbildung 2 zu sehen.

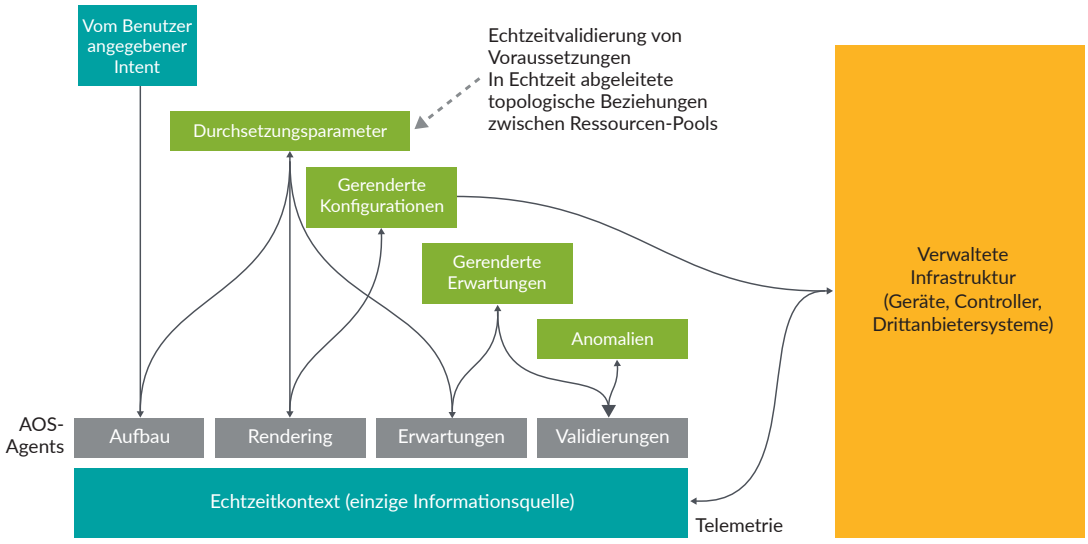

Abbildung 2: Statusbasierter Orchestrierungsablauf

Wenn Sie eine Änderung an Ihrem bestehenden, voll funktionsfähigen System vornehmen möchten, muss der Benutzer bei der zustandsbasierten Orchestrierung nur die Änderungsabsicht angeben. Dies geschieht auf eine implementierungsunabhängige Weise, was die Absichtserklärung einfacher und weniger fehleranfällig macht. Ein vollständiger Ablauf von Schritten, die während der zustandsbasierten Orchestrierung ausgeführt werden, umfasst:

Schritt 1: Echtzeit-Validierung von Voraussetzungen. Verstößt diese neue Anfrage gegen bestimmte Richtlinien? Dürfen Sie zum Beispiel dieses virtuelle Netzwerk erstellen, oder wird es Sicherheitslücken schaffen? Oder können Sie ein Gerät in den Wartungsmodus versetzen, wenn Sie wissen, dass einige andere Geräte bereits offline sind und das Abschalten dieses Geräts Ihr System angreifbar machen wird? Mit zustandsbasierter Orchestrierung werden all diese Fragen gestellt und in Echtzeit anhand des Kontextgraphen validiert.

Schritt 2: Ableitung der Parameter für die Durchsetzung in Echtzeit. Um die Absicht umzusetzen, müssen spezifische Parameter für geeignete Durchsetzungsmechanismen bereitgestellt und für bestimmte verwaltete Elemente aktiviert werden. Da Apstra ein Referenzdesign verwendet, um zu beschreiben, wie die Absicht umgesetzt werden soll, und den aktuellen Kontext versteht, kann das System automatisch die Durchsetzungsparameter in Echtzeit ableiten. Die Möglichkeit, dass ein Betreiber versehentlich einen falschen Befehl, eine falsche Schnittstelle, IP-Adresse oder ein falsches VLAN eingibt, ist ausgeschlossen.

Schritt 3: Bereitstellung mehrerer Konfigurationen in Echtzeit. Apstra führt eine mehrere Konfigurationen umfassende Bereitstellung in Echtzeit durch, die die Bereitstellung von Konfigurationen auf möglicherweise mehreren Elementen automatisiert, die über unterschiedliche hersteller- und technologiespezifische APIs verfügen können.

Schritt 4: Generieren von Erwartungen in Echtzeit. Ein Referenzdesign, das als Verhaltensstandard fungiert, ermöglicht Apstra die Generierung von Erwartungen in Echtzeit, die jene Bedingungen beschreiben, die erfüllt sein müssen, um zu bestätigen, dass das Ergebnis die Absicht erfüllt hat.

Schritt 5: Prüfen von Erwartungen in Echtzeit. Apstra veranlasst dann eine Reihe von Telemetrie- und kontextbezogenen Betriebsanalysen, um Erwartungen in Echtzeit zu prüfen.

Schritt 6: Validiertes Serviceergebnis. Am Ende dieses Prozesses kann der Benutzer ein validiertes Serviceergebnis in eindeutiger Form feststellen. Jede Änderung des Intents oder des erwarteten Betriebszustands wird im Kontextmodell in Echtzeit wiedergegeben, und jede Komponente, die von der Änderung Kenntnis haben muss, wird ebenfalls in Echtzeit darüber informiert. Apstra extrahiert relevantes Wissen aus betrieblichen Rohdaten durch absichtsbasierte Analysen.

Bei der nicht zustandsbasierten Orchestrierung fehlen viele dieser Schritte (wie in Abbildung 3 dargestellt). Das Ergebnis der Konfigurationsbereitstellung ist das, worum es bei der nicht zustandsbasierten Orchestrierung in der Regel geht: die Übertragung der Konfiguration an möglicherweise mehrere Systeme und die Überprüfung, ob die Konfigurationen von den verwalteten Elementen akzeptiert wurden. Und genau hier stößt die nicht zustandsbasierte Orchestrierung in der Regel an ihre Grenzen: Es gibt keine Definition von Serviceerwartungen oder eine Überprüfung, ob diese Erwartungen erfüllt werden. Es gibt keine automatisierte Überprüfung der Serviceergebnisse, stattdessen präsentieren die einzelnen Systeme dem Benutzer eine zentrale Übersicht des reinen Betriebszustands. Es ist Sache des Benutzers, eine visuelle Analyse vorzunehmen, um festzustellen, ob das Ergebnis erreicht wurde. Diese zentrale Übersicht enthält in der Regel zu viele Informationen und lässt den implementierungsspezifischen Kontext vermissen, was die visuelle Erkennung extrem schwierig und subjektiv macht.

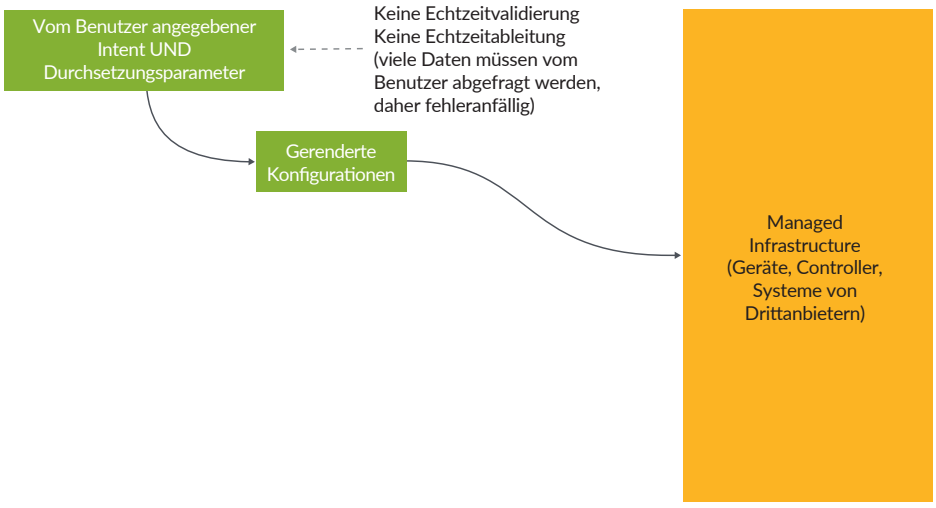

Abbildung 3: Typische nicht zustandsbasierte Orchestrierung

Absichtsbasierte Analysen (Intent-based Analytics, IBA)

Absichtsbasierte Analysen (Intent-based Analytics, IBA) unterstützen Betreiber bei der Bewältigung von Änderungen des Betriebsstatus ihrer Infrastruktur, indem sie Erkenntnisse aus Telemetrie-Rohdaten extrahieren. Wie bereits erwähnt, benötigen Sie einen in Echtzeit abfragbaren Kontext, bevor Sie absichtsbasierte Analysen verwenden können.

Die Wissensextraktion besteht aus zwei Schritten:

1. **Erkennung von wichtigen Bedingungen** (zu beobachtende Situationen). Der Nachweis der Bedingungen erfolgt durch Probes für absichtsbasierte Analysen.
2. **Automatisierung der Klassifizierung von relevanten Bedingungen und Ableitung von Beziehungen zwischen ihnen**. Die Bedingungen unterscheiden sich in ihrem semantischen Gehalt. Mit anderen Worten: Einige von ihnen sind wichtiger als andere. Es ist unerlässlich, die Beziehungen zwischen den Zuständen zu verstehen, damit Sie wichtige, handlungsrelevante Zustände (Ursachen) ausfindig machen können und verstehen, welche Zustände lediglich Folgen sind, die verschwinden, wenn wichtige Ursachen beseitigt werden. Die Klassifizierung des Zustands und die Ableitung der Kausalität erfolgen durch die Komponente der absichtsbasierten Analyse der Ursachenermittlung.

Modellieren wichtiger Bedingungen

Die absichtsbasierte Analyse modelliert die relevanten Bedingungen anhand von vier Kategorien: Anomalie, Symptom, Auswirkung und Ursache.

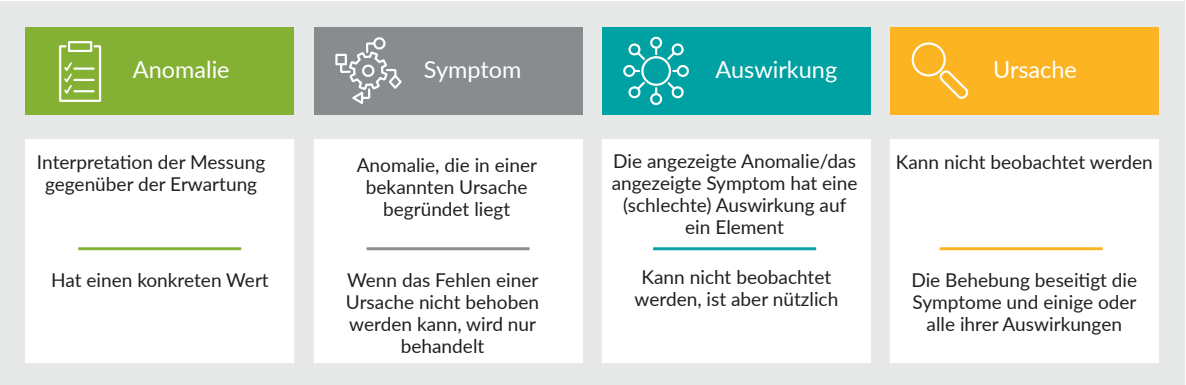

Abbildung 4: Kategorisierung der Bedingungen

Anomalien

Anomalien stellen im Wesentlichen eine Interpretation einer Messung oder eines Messwerts im Vergleich zu einer Erwartung dar und haben als solche einen greifbareren Wert als eine einfache Messung.

Symptome

Symptome sind Anomalien, die auf eine bekannte Ursache zurückzuführen sind. Sie sind in der Regel leicht zu beobachten, können aber nicht behoben, sondern nur behandelt werden, so wie die Verabreichung von Aspirin an einen Patienten zwar das Fieber, nicht aber die Krankheit selbst behandelt. Wenn Sie die Krankheit nicht behandeln, wird das Fieber zurückkehren. Wichtig ist, dass ein Symptom verschwindet, wenn die Ursache behoben ist, d. h., die Symptome sind nicht relevant, sie sind lediglich nützlich für die Diagnose der Ursache.

Auswirkungen

Auswirkungen weisen darauf hin, dass infolge einer Anomalie etwas geschehen ist. Sie sind zwar nicht immer beobachtbar, aber es ist hilfreich, sie zu kennen. So wissen Sie beispielsweise vielleicht, dass ein wichtiger Kunde von einem Geräteausfall oder einem übermäßigen Datenverlust betroffen sein wird, bevor er Sie anruft und sich beschwert.

Wenn Auswirkungen nicht feststellbar sind, können sie auf der Grundlage der Kenntnis des Intent und der Durchsetzungsmechanismen, die zur Umsetzung des Intent eingesetzt werden, berechnet werden. Das Verständnis der Auswirkungen hilft den Betreibern, Prioritäten zu setzen, welche Ursachen und Anomalien die größten Auswirkungen auf den Betrieb haben.

Ursachen

Die wichtigsten aller Bedingungen sind die Ursachen. Sie sind nicht immer erkennbar und daher schwer zu bestimmen, aber sie verursachen viele Symptome und Auswirkungen. Sie sind handlungsrelevant, da die Beseitigung der Ursache die damit verbundenen Symptome und Auswirkungen behebt.

Sondierung absichtsbasierter Analysen (IBA Probes): Wissen, was man weiß

Ein berühmtes Zitat gilt auch für Datencenter-Netzwerke: „Nicht das, was du nicht weißt, bringt dich in Schwierigkeiten, sondern das, was du sicher zu wissen glaubst, obwohl es gar nicht wahr ist.“ Der Intent definiert und konkretisiert das Normale – das, was wir wissen. Die absichtsbasierte Analyse (IBA) stellt sicher, dass das, was wir wissen, auch tatsächlich so ist, wie es sein sollte.

> *„Nicht das, was du nicht weißt, bringt dich in Schwierigkeiten, sondern das, was du sicher zu wissen glaubst, obwohl es gar nicht wahr ist.“* – Mark Twain

Die Sondierung absichtsbasierter Analysen (IBA Probe) ist für die Erkennung relevanter Bedingungen verantwortlich und wird durch einen Verhaltensstandard gesteuert, der Teil des Referenzdesigns ist. Die Sondierung absichtsbasierter Analysen erfasst Daten, verarbeitet sie und vergleicht dann das Ergebnis mit den Erwartungen. Die Sondierung einer absichtsbasierten Analyse ist im Wesentlichen eine konfigurierbare Datenverarbeitungs-Pipeline, die es den Benutzern ermöglicht, relevante Bedingungen (d. h. zu beobachtende Situationen) festzulegen.

Datenquellen und -abfragen

Die ersten Phasen einer Sondierung sind in der Regel Datenquellen, die für das Abrufen der Telemetrie-Rohdaten zuständig sind. Die Konfiguration der Datenquellenphase umfasst eine Abfrage, die genau angibt, welche Daten gesammelt werden sollen – eine sehr leistungsstarke Funktion.

Ein Beispiel: Ein Betreiber möchte ein Ungleichgewicht beim Equal-Cost-Multipath-Routing (ECMP) analysieren, das nur für Fabric-Schnittstellen gilt, und es gibt Berichte, dass eine bestimmte Betriebssystemversion einen Fehler im ECMP-Hashing-Algorithmus eingeführt hat. Eine Abfrage kann die Notwendigkeit zum Ausdruck bringen, Schnittstellenzähler auf jedem Top-of-Rack-Switch zu erfassen, aber nur für Fabric-Schnittstellen und für Switches, auf denen eine bestimmte Version x.y.z des Switch-Betriebssystems läuft. Da diese zu überwachende Situation nun festgelegt ist, muss sich der Betreiber nicht mehr um die Änderungen kümmern. Wenn ein neuer Fabric-Link zu einem Switch hinzugefügt wird, der die Kriterien erfüllt, wird er automatisch in die Analyse einbezogen. Wenn ein neuer Switch hinzugefügt wird, wird er automatisch einbezogen. Wenn jemand einen anderen Switch auf Version x.y.z aktualisiert, wird er automatisch einbezogen. Die Sondierung der absichtsbasierten Analyse (IBA Probe) erfordert keine Wartungsmaßnahmen.

Phasenverarbeiter

In vielen Fällen ist der Betreiber nicht an Echtzeitwerten von Telemetrie-Rohdaten interessiert, sondern eher an einer Aggregation oder an Trends. Die absichtsbasierte Analyse enthält Phasenverarbeiter, die Informationen aggregieren, wie z. B. die Berechnung von Durchschnitt, Min/Max, Standardabweichung usw. Ein Betreiber kann dann diese Aggregate mit den Erwartungen abgleichen, um festzustellen, ob die zusammengefasste Metrik innerhalb oder außerhalb eines bestimmten Bereichs liegt. In diesem Fall wird eine Anomalie gemeldet.

Der Betreiber kann dann prüfen, ob diese Anomalie über einen Zeitraum anhält und einen bestimmten Schwellenwert überschreitet, und kann die Anomalie nur dann kennzeichnen, wenn der Schwellenwert überschritten wird, um die Kennzeichnung von Anomalien für vorübergehende oder temporäre Bedingungen zu vermeiden. Dazu kann der Betreiber einfach eine nachfolgende Stufe so konfigurieren, dass sie einen so genannten Time-in-State-Prozessor enthält.

Mehr als nur numerische Daten

Sondierungen arbeiten nicht nur mit numerischen Daten. Sie können unter anderem dazu verwendet werden, die Richtigkeit der Control- und Data Plane zu überprüfen. Ein Betreiber kann z. B. einen Datasource-Prozessor mit einer Abfrage konfigurieren, um eine erwartete Routing- oder Weiterleitungstabelle zu erstellen, die dem Intent entspricht.

In einer Ethernet-VPN (EVPN) Umgebung enthält die Zielsetzung Informationen wie die vorhandenen virtuellen Netzwerke, die Standorte ihrer Endgeräte und Informationen über Durchsetzungsmechanismen. Diese erwartete Tabelle kann dann mit der aus der Telemetrie stammenden Tabelle verglichen werden, und Anomalien werden markiert, wenn eine Abweichung festgestellt wird.

Eine Probe kann auch so konfiguriert werden, dass sie plötzliche Änderungen in den Größen von Weiterleitungs- und Routing-Tabellen verfolgt und eine Warnung ausgibt, wenn die Trends nicht den Erwartungen entsprechen. Der „erwartete“ Schwellenwert kann auch dynamisch anhand des Intent berechnet werden, da er eine Funktion der Anzahl der virtuellen Netzwerke, der Anzahl der Endgeräte, der Anzahl der virtuellen Tunnelendgeräte (VTEPs), der Anzahl der VTEPs mit bestimmten Problemen usw. sein kann. Die Möglichkeiten sind unbegrenzt.

Sondierungen können per einfachem REST-Aufruf oder über eine grafische Benutzeroberfläche deklarativ erstellt, aktiviert und deaktiviert werden. Die Aktivierung der Sondierung dient auch als Auslöser für die Telemetrie. Mit anderen Worten, bestimmte Telemetriedaten werden nur dann erfasst, wenn eine Sondierung daran interessiert ist, sodass die Sondierungen im Wesentlichen als Konfigurationsmechanismus für die Telemetrieerfassung dienen. Abfragen in Datasource-Prozessoren ermöglichen eine granulare und präzise Konfiguration, sodass eine „Beeinträchtigung durch Datensammlung“ vermieden wird, die auftritt, wenn Unmengen von Daten gesammelt werden, ohne eine klare Vorstellung davon zu haben, was damit geschehen soll. Dies wiederum treibt die Kosten für die Speicherung und Verarbeitung von Daten in die Höhe.

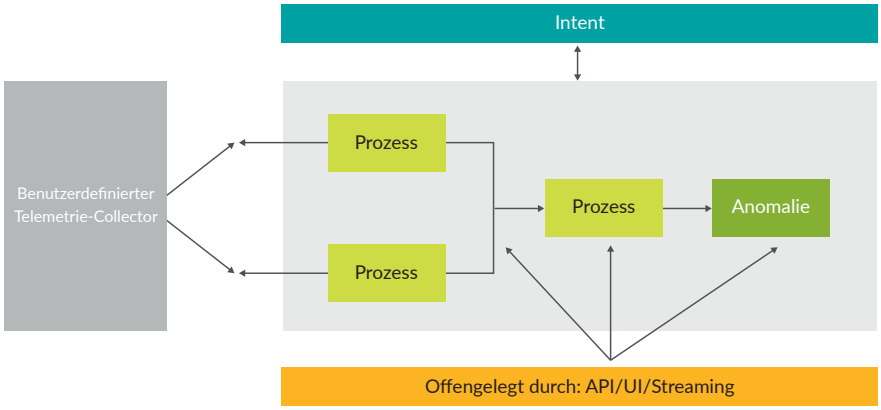

Abbildung 5: Telemetrieerfassung

Sobald eine Sondierung konfiguriert ist, ist die Ausgabe jeder Phase über API, GUI und Streaming-Endgeräte verfügbar. Apstra umfasst eine Reihe interner Sondierungen, und es gibt ein Repository von Open-Source-Probes auf GitHub. Darüber hinaus können Benutzer neue Sondierungen erstellen.

Ursachenermittlung

Die Ursachenermittlung (Root-Cause-Identification, RCI) ist ein Mechanismus zur Automatisierung der Klassifizierung und Ableitung von Kausalitätsbeziehungen zwischen den in der Infrastruktur festgestellten Zuständen. Der Hauptvorteil der RCI besteht darin, dass sie aus der Vielzahl an nicht handlungsrelevanten Zuständen, die lediglich Folgen von Ursachenfehlern sind, die Bedingungen aufzeigt, die ein Eingreifen des Betreibers erfordern. Nehmen wir zum Beispiel an, ein Betreiber hat die Infrastruktur gut instrumentiert und beobachtet ein Protokoll anomaler Bedingungen in einer „zentralen Übersicht“ (siehe Abbildung 6). Die roten Punkte zeigen verschiedene Arten von Bedingungen an, die über die gesamte Infrastruktur verstreut sind und in vielen verschiedenen Elementen auftreten.

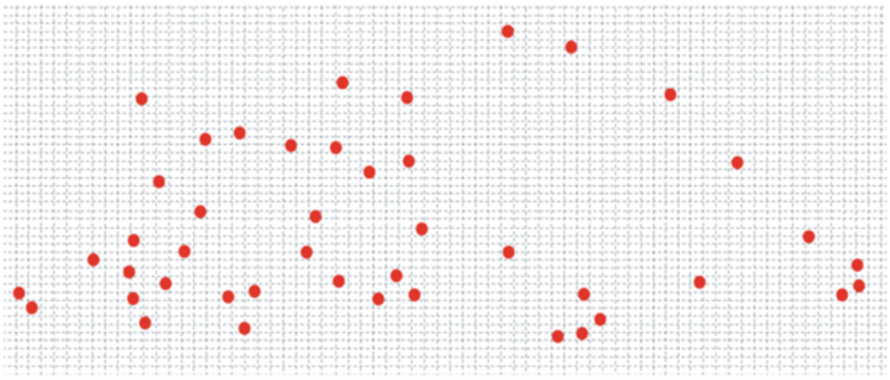

Abbildung 6: Konsole mit einer zentralen Übersicht

Das Problem bei diesem Bild ist, dass es keine Handlungsmöglichkeiten eröffnet – es ist einfach nur eine Ansammlung von Daten. Ohne Klassifizierung wetteifert jede Bedingung um die Aufmerksamkeit des Bedieners, aber auf welche muss er reagieren? Hier kommt RCI ins Spiel. Ausgestattet mit dem Wissen über den Kontext, der besagt, was der Service ist, wie er implementiert und in Durchsetzungsmechanismen abgebildet wird und wie die Elemente in der verwalteten Infrastruktur zusammenhängen, identifiziert und klassifiziert RCI die Ursache(n) und die damit verbundenen Symptome und Auswirkungen. Abbildung 7 zeigt die Ergebnisse der Analyse mit RCI:

- Die Hauptursache war ein Speicherausfall auf einem Switch namens „spine_1“ (Ursache: großer blauer Punkt).
- Dieser Speicherausfall führte dazu, dass der Prozesskiller bei einigen Prozessen aktiv wurde und diese beendete, darunter auch ein Routing-Prozess (Symptome: dunkelblaue Punkte).
- Dies wiederum führte dazu, dass BGP-Sitzungen auf diesem und den Peering-Geräten fehlschlugen und erwartete Routing-Tabelleneinträge fehlten (Symptome: dunkelblaue Punkte).
- Infolgedessen kam es bei den Endgeräten der Kunden X und Y zu Verbindungsproblemen (Auswirkungen: hellblaue Punkte).

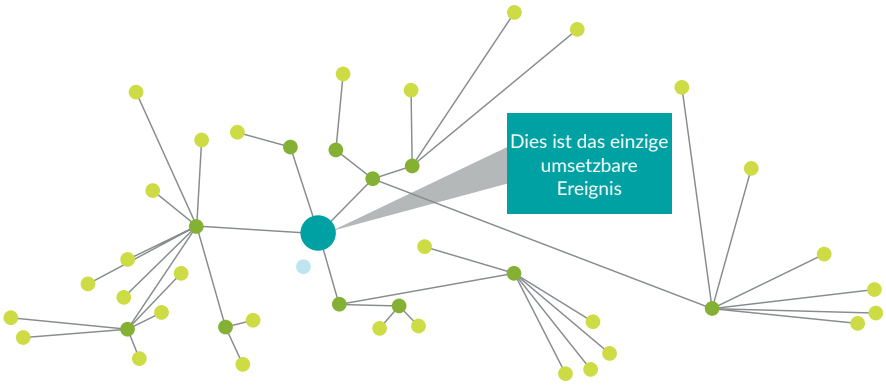

Abbildung 7: Konsole zur Ursachenermittlung

RCI automatisiert den komplexen Denkvorgang, der mit einer „zentralen Übersicht“ verbunden ist, die die Betreiber mit Bergen von Informationen überfordert und es ihnen überlässt, auf visuelle Problementdeckung zu gehen und Korrelationen herzustellen. Stattdessen liefert die RCI dem Betreiber eine einfache Übersicht, die die erforderlichen Maßnahmen aufzeigt.

Kontextmodell

Wenn man alle Daten hat, heißt das nicht, dass man auch alle Antworten hat. Und das Sammeln riesiger Datenmengen ohne Automatisierung der Informationsextraktion führt zwangsläufig zu einer Explosion der Betriebskosten, da Ihre Fachkräfte wertvolle Zeit damit verbringen, aus den Daten einen Sinn abzuleiten. Daten an sich haben nur einen begrenzten Wert, es sei denn, sie vermitteln Ihnen Erkenntnisse oder beantworten die richtigen Fragen.

Telemetrie-Rohdaten werden in der Regel in Schlüssel-Wert-Speichern hinterlegt, die sich gut für horizontale Skalierung und Sharding eignen. Außer einfachen Schlüsselabfragen unterstützen sie jedoch keine weiteren Abfragen. Die Erkenntnisextrahierung erfordert Unterstützung für die Abfragen sowie einen verfügbaren Kontext, der das Erstellen der Abfragen ermöglicht.

Andererseits unterstützen SQL-Datenspeicher auch komplexe Abfragen. Das Problem besteht darin, dass SQL-Tabellen für erwartete Abfragen konzipiert sind, die während des Entwurfs erstellt werden. Wenn Sie während der Ausführung eine andere Frage stellen möchten, kann es sein, dass Sie die Datenbank entnormalisieren oder dass die Abfragen mit vielen Joins enden, was die Leistung zum Erliegen bringt. Bei der Extraktion von operativem Wissen geht es jedoch um Abfragen, die während der Ausführung auftauchen. Dies ist der Punkt, an dem die graphbasierte Implementierung von Kontextdaten hervorsticht: Sie unterstützt beliebige komplexe Abfragen mit einer großen Anzahl von „Joins“ während der Ausführung. Das Graphmodell kann auch problemlos erweitert werden, indem einfach weitere Instanzen von Grundbausteinen, Knoten und Beziehungen hinzugefügt werden.

Sehen wir uns genauer an, was in diesem Zusammenhang eine Rolle spielt, denn das erklärt die Bedeutung, die Abfragen entfalten.

- **Designartefakte**: Zunächst enthält der Intent-Graphkontext Designartefakte. Wenn Sie beispielsweise ein Netzwerk für ein Datencenter entwerfen, kann es den gewünschten Überbelegungsfaktor, die gewünschte Anzahl von Servern und die gewünschte Hochverfügbarkeitskonfiguration (single-attached, dual-attached) enthalten.
- **Ressourcenzuweisung**: Dies umfasst Entscheidungen über die Zuweisung von Ressourcen. Sie bestimmt, ob Sie Ihre Infrastruktur als „Nutztier“ oder als „Schoßtier“ verwalten.
- **Isolationsrichtlinien**: Darin werden auch die Isolationsrichtlinien festgelegt. Ist es erlaubt, bestimmte Ressourcen (IPs, ASNs, VLANs) wiederzuverwenden?
- **Segmentierungsrichtlinien**: Diese enthält Segmentierungsrichtlinien, die festlegen, welche Endgeräte und Workloads miteinander kommunizieren können und unter welchen Bedingungen.
- **Informationen zur Durchsetzung**: Diese enthalten Durchsetzungsinformationen darüber, wo (auf welchen physischen oder virtuellen Appliances) und wie (mit welchen Mechanismen) die Segmentierungs- und Erreichbarkeitsrichtlinien tatsächlich umgesetzt wurden.
- **Richtlinien zur externen Erreichbarkeit**: Im Fall des Datencenters kann dies auch Richtlinien zur externen Erreichbarkeit enthalten, die angeben, welche internen Endgeräte für die Außenwelt sichtbar sein dürfen und umgekehrt. Sie besagen auch, welche Mandanten welche Segmente besitzen, welche Servicelevels jedem von ihnen zugesagt wurden und welche Servicelevel-Ziele verfolgt werden.
- **Geschäftsperspektive**: Dabei geht es um die geschäftliche Perspektive, z. B. um die Servicelevel-Vereinbarungen und die Kosten, die entstehen, wenn Sie diese nicht einhalten.

8-D-Ansicht

Auch wenn alle diese Artefakte in einem einzigen Graphen dargestellt werden, ist diese Single Source of Truth logischerweise ein einziger mehrdimensionaler Raum mit mindestens den folgenden acht Dimensionen (die speziell aus diesem Beispiel abgeleitet wurden; es könnten auch noch mehr sein):

- Design
- Ressourcen
- Isolation
- Segmentierung
- Durchsetzung
- Externe Erreichbarkeit
- Servicelevels
- Geschäftsperspektive

Wenn also etwas schief läuft, können Sie mit einer einzigen Abfrage diese 8-D-Ansicht untersuchen und komplexe Fragen beantworten, wie zum Beispiel: „Sind meine Designziele angesichts des jüngsten Fehlers bei einem Element noch gültig?“, „Werden die richtigen Ressourcen an den richtigen Durchsetzungspunkten mit den gewünschten Isolierungsrichtlinien verwendet?“, „Wirkt sich dies in irgendeiner Weise auf die Segmentierungsrichtlinie aus?“, „Wird das Servicelevel-Ziel eingehalten?“, „Welchen Preis zahle ich für diesen Fehler?“

Diese 8-D-Ansicht ist kein „unbedeutendes Extra“, sondern Voraussetzung für den zuverlässigen Betrieb. Bei dem absichtsbasierten Apstra-System ist die 8-D-Ansicht integriert. Ohne Apstra müssen Sie diese 8-D-Ansicht auf einer sehr komplexen Ebene rekonstruieren, die mehrere Informationsquellen und Wissensbestände umfasst, die in den Köpfen Ihrer Experten existieren. Der Aufbau dieser Ebene in einer Umgebung, in der die einzelnen Informationsquellen nicht mit Blick auf die Integration entwickelt wurden und unterschiedliche Semantiken und Verhaltensweisen aufweisen, stellt im besten Fall eine schwerfällige und undifferenzierte Aufgabe und im schlimmsten Fall einen unüberschaubaren Albtraum dar.

Überwachung und Benachrichtigungen in Echtzeit

Bei Apstra geht es darum, Erkenntnisse über den Intent und den daraus resultierenden Betriebszustand Ihrer Infrastruktur zu gewinnen. Dieses Wissen ist machtvoll, wenn es zeitnah ist, d. h. wenn es die aktuellen Bedingungen widerspiegelt. Das Herzstück von Apstra ist die Möglichkeit, Live-Abfragen zu konfigurieren, die es Kunden ermöglichen, Bedingungen von Interesse zu abonnieren und in Echtzeit benachrichtigt zu werden, wenn die Bedingungen erfüllt sind. Die Bedingungen beziehen sich in diesem Zusammenhang sowohl auf den Intent als auch auf den Betriebszustand. Dies ist eine unabdingbare Voraussetzung für die zuverlässige Bearbeitung von Änderungen.

Wie Sie im Abschnitt Überblick über die Architektur sehen werden, wird das gleiche Publish-Subscribe-Paradigma, das auf der Benutzerebene präsentiert wird, durch einen äquivalenten Low-Level-Publish/Subscribe-Mechanismus unterstützt. Dieser Mechanismus wird dadurch realisiert, dass der verteilte Datenspeicher als datenzentrierter logischer Kommunikationskanal für die Apstra-Systemprozesse dient, die die Anwendungslogik implementieren.

Selbstständiger Betrieb

Nicht zuletzt besteht die Notwendigkeit, die Reaktion auf bestimmte Ereignisse zu automatisieren, um Probleme zu beheben, sie für forensische Analysen zu protokollieren oder einen Drilldown auf der nächsten Ebene durchzuführen, um Telemetriedaten zu sammeln, die bei der Ursachenanalyse helfen.

Eine automatisierte Reaktion:

- Reduziert das Risiko, da Korrekturparameter automatisch aus der aktuellen Single Source of Truth abgeleitet werden und nicht von Fehlkonfigurationen oder veralteten Daten abhängig sind.
- Verbessert die Kundenerfahrung, da die Abhilfemaßnahmen zeitnah erfolgen.
- Reduziert die Betriebskosten, da die manuelle Fehlerbehebung und die manuelle Ausführung des Abhilfeplans entfallen.

Die Hürden für ein selbstständig arbeitendes Netzwerk ergeben sich eher aus organisatorischen oder individuellen Widerständen als aus technologischen Beschränkungen, da die erforderlichen Funktionen bereits vorhanden sind.

Letztlich ermöglicht eine automatisierte Reaktion den selbstständigen Betrieb und die selbstständige Reparatur des Netzwerks. Die Hürden für ein selbstständig arbeitendes Netzwerk ergeben sich eher aus organisatorischen oder individuellen Widerständen als aus technologischen Beschränkungen, da die erforderlichen Funktionen bereits vorhanden sind.

Erweiterbarkeit: Zukunftsfähiges Datencenter-Netzwerk

Die Erweiterbarkeit hat drei Dimensionen:

1. **Erweiterbarkeit des Referenzdesigns**. Sie regelt, wie die einzelnen Teile der Infrastruktur zusammenarbeiten, um das Ziel zu erreichen. Es umfasst Plug-ins, die Änderungen des Graphmodells sowie Änderungen im Zusammenhang mit der Übertragung der Absicht auf Durchsetzungsmechanismen in der Infrastruktur ermöglichen.
2. **Erweiterbarkeit der Analyse**. Dies bezieht sich auf die Definition neuer Bedingungen oder Situationen, die beobachtet werden sollen, und ermöglicht es dem Benutzer, neue Überprüfungen zu definieren und festzulegen, wie sie klassifiziert und in Beziehung gesetzt werden sollen.
3. **Flexible Service-APIs**. Diese bieten die Möglichkeit, die Servicedefinitionen der obersten Ebene zu erweitern.

Referenzdesign

Wie bereits erwähnt, ist das Referenzdesign ein Verhaltensstandard, der definiert, wie der Intent auf die Durchsetzungsmechanismen abgebildet wird und welche Erwartungen vorgegeben sind, damit das Ziel als erfüllt gilt. Jeder Aspekt dieses Standards kann geändert oder erweitert werden. Es können neue Knoten- und Beziehungstypen definiert sowie Konfigurationsvorlagen und Ressourcenzuweisungsmechanismen geändert werden.

Erweiterbarkeit der Analyse

Es gibt zwei Mechanismen, um neue Analysefunktionen einzuführen:

1. Es können **neue Sondierungen für absichtsbasierte Analysen** definiert werden, die neue relevante Bedingungen ermitteln. Sie können neue Telemetriekollektoren sowie zustandsspezifische Datenverarbeitungspipelines umfassen. Sondierungen können auch veröffentlicht und anschließend aus einem öffentlichen Repository importiert werden.
2. **Neue RCI-Modelle** können definiert und in ein System geladen werden. Ein RCI-Modell ist im Wesentlichen eine Zuordnung einer neuen Art von Ursache zu einer Reihe von Symptomen, die sie erzeugt. Sobald dieses Modell definiert und geladen ist, automatisiert das Apstra-System die Identifizierung der Ursache auf der Basis der beobachteten Symptome.

Service-APIs

Apstra stellt APIs auf Dienstebene in Form von gruppenbasierten Richtlinien zur Verfügung und bietet so die Flexibilität, eine breite Palette von Services und Richtlinien in einer implementierungsunabhängigen Weise zu unterstützen.

Bei gruppenbasierten Richtlinien wird die Absicht in Form eines Graphs ausgedrückt, der Endgeräte darstellt, die in Gruppen (Mitgliederbeziehungen) eingeordnet werden, um den Intent für ein bestimmtes gemeinsames Verhalten auszudrücken. Richtlinien werden instanziiert und mit Gruppen oder einzelnen Endgeräten verknüpft, um dieses Verhalten zu definieren.

Richtlinien können sich auf eine Gruppe in einer direktionalen (von/bis-Beziehung) oder nicht-direktionalen (gilt für) Weise beziehen. Richtlinien sind eine Sammlung von Regeln. Regeln können eine Next-Rule-Beziehung haben, wenn die Reihenfolge zwischen den Regeln wichtig ist. Gruppen können aus anderen Gruppen zusammengesetzt sein (Hierarchie). Gruppen können auch eine Beziehung zu anderen Gruppen haben, um bestimmte Einschränkungen auszudrücken (z. B. „diese Endgeräte/Gruppen befinden sich hinter diesen Portgruppen“). Endgeräte, Gruppen, Richtlinien und Regeln können als Bausteine betrachtet werden, um den Konnektivitäts-Intent auszudrücken.

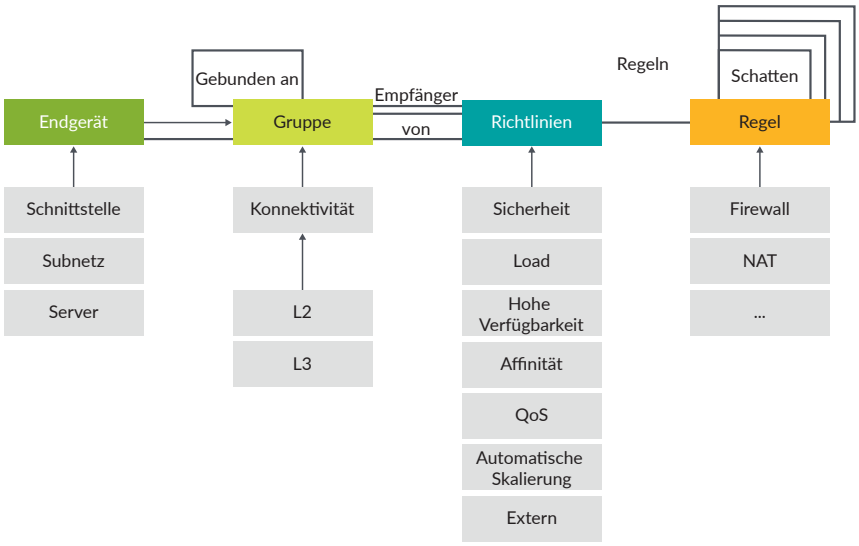

Abbildung 8: Gruppenbasierte Richtlinien

Endgeräte sind Elemente Ihrer Infrastruktur, für die Richtlinien gelten, und als solche sind sie recht allgemein gehalten. Sie können eine Schnittstelle (physisch oder virtuell), einen Server, eine virtuelle Maschine (VM), einen Container oder ein Anwendungsendgerät darstellen. Die blauen Pfeile in Abbildung 8 zeigen die „logische“ Verknüpfung an. Endgeräte enthalten Parameter, die sie genauer definieren (z. B. Schnittstellenname, Portnummer, Seriennummer und Hostname, VM UUID, Container-IP, Anwendungsprotokoll, UDP/TCP-Portnummer). Endgeräte können auch Elemente darstellen, die nicht von Apstra verwaltet werden (externe Endgeräte), und als solche werden sie verwendet, um Beschränkungen von externen Systemen auszudrücken, mit denen Apstra interagieren muss (z. B. externe Router-IP/ASN).

Endgeräte werden in Gruppen eingeteilt. Jedes beliebige Endgerät kann Mitglied mehrerer Gruppen sein, die jeweils unterschiedliche Aspekte des beabsichtigten Verhaltens zum Ausdruck bringen. So könnte eine Gruppe zum Beispiel angeben:

- Eine Auswahl von Servern in der Region „Ost“
- Eine Gruppe von Servern, die für das Load Balancing verwendet werden sollen
- Eine Reihe von Servern, die eine Redundanzgruppe bilden

Gruppen können auch eine überladene Semantik haben. So sind beispielsweise Endgeräte, die Mitglieder der L2-Domänengruppe sind, Mitglieder einer L2-Broadcast-Domäne (Subnetz). Gleichermaßen haben Endgeräte, die Mitglieder der L3-Domäne sind, eine L3-Erreichbarkeit.

Gruppen können auch aus anderen Gruppen zusammengesetzt sein. Endgerätemitglieder der Gruppe können explizit instanziiert werden, oder es kann eine dynamische Mitgliedschaftsspezifikation als Teil einer Gruppendefinition geben, bei der Endgeräte von der in der Gruppe vorhandenen Spezifikation impliziert werden.

Eine L3-Domäne kann zum Beispiel Classless Interdomain Routing (CIDR) oder ein Subnetz als Eigenschaft haben. Daher sind alle Endgeräte mit IPs in diesem Bereich/Subnetz implizit Mitglieder der Gruppe. In diesem Fall spezifiziert der L3-Referenzentwurf für den Server eine Reihe von externen/internen Teilnetzen. Auch wenn die Endgeräte (Container) in Apstra nicht explizit angegeben und verwaltet werden, wird vorausgesetzt, dass die Container zu den angegebenen L3-Domänen gehören. Um jedoch eine granulare Segmentierung zu erreichen, werden wir auch explizite Spezifikationen für Container-Endgeräte und die Server, auf denen sie gehostet werden, einführen.

Eine Richtlinie definiert das allgemeine Verhalten und kann Parameter enthalten, um dieses Verhalten genau festzulegen. Richtlinien können nicht-direktional auf die Gruppe angewandt werden, oder die Richtung kann bei Bedarf angegeben werden, wie z. B. bei Sicherheitsrichtlinien. Beispiele für spezifische Richtlinien sind Sicherheit, Load Balancing, Hohe Verfügbarkeit, Affinität (z. B. der Wunsch, Endgeräte zusammenzulegen oder zu verteilen) und Quality of Service (QoS).

Richtlinien können bei Bedarf Regeln enthalten. Regeln folgen normalerweise dem Muster „Bedingung gefolgt von einer Aktion“. In der Sicherheitsrichtlinie ist „Übereinstimmung“ beispielsweise eine Bedingungsanweisung und „Aktion“ ist „Zulassen/Ablehnen/Protokollieren“.

Skalierbarkeit: Problemlos wachsen

Apstra unterstützt den netzwerktransparenten Zugriff auf verteilte Zustände und deren Verwaltung, während die parallele Ausführung durch separate Prozesse unterstützt wird. Die Echtzeitausführung wird durch ein ereignisgesteuertes asynchrones Ausführungsmodell in Verbindung mit einer Echtzeitplanung der Ausführung unterstützt. Effizienz und Vorhersagbarkeit werden durch Kompilierung mit C++ als Zwischensprache unterstützt, um Effizienz auf Computerebene zu erreichen. Es gibt drei Dimensionen für die Skalierung.

Skalierung des Zustands.

Die erste Dimension der Skalierung ist der Zustand. Apstra-Datenspeicher können horizontal skaliert werden, indem weitere Serverpaare mit hoher Verfügbarkeit (HA) hinzugefügt werden. Intent- und Telemetriedatenspeicher sind getrennt und können bei Bedarf unabhängig voneinander skaliert werden. Es gibt auch eine Bestimmung für hierarchische Datenspeicher, bei der ein Datenspeicher der obersten Ebene nur die erforderliche (vom Entwickler festgelegte) Teilmenge des Zustands aus den Datenspeichern der nächsten Ebene bezieht (synchronisiert), die für die Korrelation des Zustands zwischen den Datenspeichern erforderlich ist.

Skalierung der Verarbeitung

Die zweite Dimension der Skalierung ist die Verarbeitung. Apstra kann bei Bedarf mehrere Kopien von Verarbeitungsagenten (pro Agententyp) starten, die sich die Verarbeitung teilen. Weitere Agenten können durch Hinzufügen weiterer Server ergänzt werden, auf denen sie gehostet werden. Apstra verwaltet auch den Lebenszyklus der Agenten.

Die zustandsbasierte Publish--Subscribe-Architektur des Systems ermöglicht es Agenten, auf eine genau definierte Teilmenge von Zuständen zu reagieren (Anwendungslogik bereitzustellen). Der gesamte Intent wird durch separate Agenten abgedeckt, die mit der Bearbeitung verschiedener Teilbereiche des Zustands beauftragt werden. Das bedeutet, dass der Agent bei einer Änderung des Intent oder des Betriebszustands mit einer „inkrementellen Änderung“ reagiert, die unabhängig vom Umfang des gesamten Zustands ist.

Apstra verwendet den traditionellen Ansatz zur Bewältigung des Umfangs und der damit verbundenen Komplexität: Zerlegung. Der Ansatz „jeder weiß alles" ist nicht praktikabel. Man muss das Wissen über den gewünschten Zustand weiterleiten und jeden Agenten entscheiden lassen, wie dieser Zustand erreicht werden soll, damit keine zentrale Entscheidung getroffen werden muss. Die Apstra-Unterstützung für Live-Graphabfragen bedeutet, dass Clients wie die Benutzeroberfläche genau das anfordern können, was sie wollen, und genau das bekommen, was sie brauchen – und nicht mehr –, sodass die Datenmenge, die vom Backend abgerufen werden soll, genau festgelegt werden kann.

Skalierung von Netzwerkdatenverkehr

Die dritte Dimension ist die Skalierung des Netzwerkdatenverkehrs. Die Kommunikation zwischen den Agenten und dem Datenspeicher erfolgt über einen optimierten Binärkanal, wodurch der Datenverkehr im Vergleich zu textbasierten Protokollen erheblich reduziert wird.

Fehlertoleranz wird erreicht, indem eine Apstra Anwendung als mehrere Prozesse ausgeführt wird, die möglicherweise auf separaten, über ein Netzwerk verbundenen Hardwaregeräten laufen, und indem der Zustand von der Verarbeitung getrennt wird, wobei ein replizierter Zustand und eine schnelle Wiederherstellung des Zustands unterstützt werden.

Apstra-Architekturübersicht

Im ersten Teil dieses Whitepapers haben wir einige der Herausforderungen der Netzwerkarchitektur von Datencentern erörtert und wie Apstra diese Probleme löst. Nun gehen wir auf die Besonderheiten der Architektur von Apstra ein.

Apstra basiert auf einer verteilten Zustandsverwaltungsinfrastruktur, die als datenzentrierte Kommunikations-Fabric mit horizontal skalierbarem und fehlertolerantem In-Memory-Datenspeicher beschrieben werden kann. Alle Funktionalitäten der spezifischen Referenzdesign-Anwendung werden über eine Reihe Stateless Agenten implementiert. Agenten kommunizieren miteinander über einen logischen, auf Publish-Subscribe basierenden Kommunikationskanal und implementieren im Wesentlichen die Logik der Anwendung.

Jede Apstra Referenzdesignanwendung ist einfach eine Sammlung von zustandslosen Agenten, wie oben beschrieben. Im Grunde genommen gibt es drei Klassen von Agenten:

1. **Interaktionsagenten (Webagenten)** sind für die Interaktion mit den Benutzern zuständig, d. h. sie empfangen Benutzereingaben und versorgen die Benutzer mit relevantem Kontext aus dem Datenspeicher.
2. **Anwendungsagenten** sind für die Durchführung anwendungsdomänenspezifischen Datentransformationen zuständig, indem sie Eingabeeinheiten abonnieren und Ausgabeeinheiten erzeugen.
3. **Geräteagenten** befinden sich auf einem verwalteten physischen oder virtuellen System (oder sind Proxys für dieses), z. B. auf einem Switch, Server, einer Firewall, einem Load Balancer oder sogar einem Controller. Sie werden zum Schreiben der Konfiguration und zum Sammeln von Telemetriedaten über native (gerätespezifische) Schnittstellen verwendet.

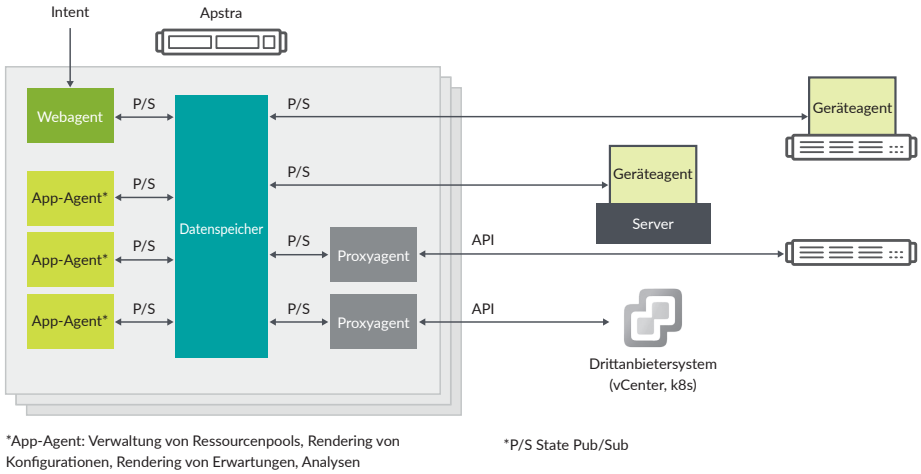

Abbildung 9: Apstra Agenten

Diese Interaktion kann anhand eines Beispiels veranschaulicht werden, das einen Teil der Apstra Referenzanwendung für Netzwerke in Datencentern beschreibt.

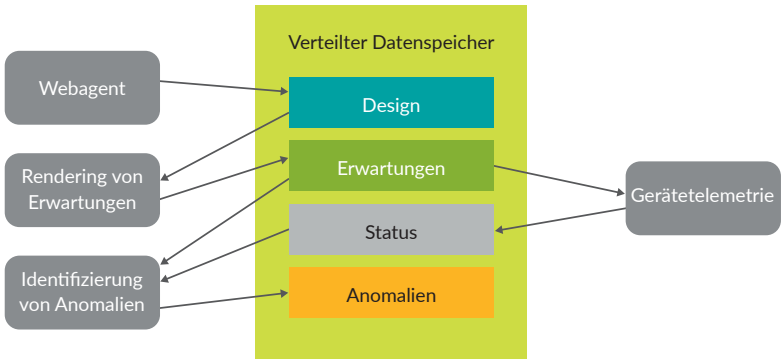

Abbildung 10: Apstra Intent-based System – Referenzanwendung für Datencenter-Netzwerke

Der Webagent empfängt Benutzereingaben – in diesem Fall einen Entwurf für eine L3-Clos-Fabric, der die Anzahl der Spines, Leaves und Links zwischen ihnen sowie die Ressourcenpools für Fabric-IPs und ASN-Nummern (Abstract Syntax Notation) enthält. Der Webagent veröffentlicht diesen Intent im Datenspeicher als eine Reihe von Graphknoten und -beziehungen sowie deren jeweilige Eigenschaften.

Der Build-Agent abonniert diesen Intent und:

- prüft die Korrektheit und Vollständigkeit
- weist Ressourcen aus Ressourcenpools zu

Wenn die Überprüfungen erfolgreich waren, veröffentlicht der Build-Agent diese Absicht zusammen mit den Ressourcenzuweisungen im Datenspeicher:

1. Der Konfigurations-Rendering-Agent bezieht die Ausgabe des Build-Agenten.
2. Für jeden Knoten ruft der Konfigurationsagent die relevanten Daten, einschließlich der Ressourcen, ab und fügt sie mit Konfigurationsvorlagen zusammen.
3. Der Agent für Erwartungen bezieht ebenfalls die Ausgabe des Build-Agenten und erzeugt Erwartungen, die erfüllt werden müssen, um das Ergebnis zu validieren.
4. Der Agent für Gerätetelemetrie bezieht die Ausgabe des Agenten für Erwartungen und beginnt mit der Erfassung relevanter Telemetriedaten.
5. Sondierungen für absichtsbasierte Analysen verarbeiten die rohen Telemetriedaten, vergleichen sie mit den Erwartungen und veröffentlichen Anomalien.
6. Der RCI-Agent analysiert die Anomalien und klassifiziert sie in Symptome, Auswirkungen und identifizierte Ursachen.

Agenten kommunizieren über attributbasierte Schnittstellen (daher der Begriff datenbasiert), indem sie Einheiten veröffentlichen und Änderungen an Einheiten verfolgen. Datenbasiert bedeutet auch, dass die Datendefinition Teil des Frameworks ist und durch die Definition der Einheiten umgesetzt wird, im Gegensatz zu z. B. nachrichtenbasierten Systemen.

Das datenbasierte Publish-Subscribe-System leidet nicht unter den Problemen der nachrichtenbasierten Systeme. In einem nachrichtenbasierten System übersteigt die Anzahl der Nachrichten früher oder später die Kapazität des Systems, sie zu speichern oder zu verarbeiten. Der Umgang damit ist schwierig, da man den Verlauf der Nachrichten erneut nachvollziehen muss, um einen konsistenten Zustand zu erreichen.

Das datenbasierte System ist unabhängig von schnellen Zustandsänderungen, da es im Grunde nur vom letzten Zustand abhängt. Dieser Zustand erfasst den wichtigen Kontext und abstrahiert alle möglichen (und irrelevanten) Ereignisabläufe, die zu ihm geführt haben. Code, der mit dem Paradigma des Zustandssystems geschrieben wurde, ist einfacher zu lesen, zu warten und zu debuggen.

Schwierige Probleme (z. B. Flexibilität und Fehlertoleranz) werden einmalig und im Namen aller Agenten gelöst. Die typische Architektur besteht dann aus einer Reihe von zustandslosen Agenten, die im Falle eines Fehlers neu gestartet werden können und dort weitermachen, wo sie aufgehört haben, indem sie einfach den Zustand, den sie in der Systemdatenbank (SysDB) bezogen haben, neu lesen.

Vorteile der Apstra-Architektur

Die Apstra-Architektur bietet erhebliche Vorteile und löst einige der schwierigsten Netzwerkprobleme in Datencentern, wie wir in diesem Dokument beschrieben haben. Die Architektur:

- **Hilft Betreibern, zuverlässig mit Veränderungen umzugehen**. Dies ist durch in Echtzeit abfragbare Intents und den Betriebskontext möglich.
- **Simplifiziert alle Aspekte des Lebenszyklus von Netzwerkservices**, einschließlich der Vorgänge an Tag 0, 1 und 2. Und Einfachheit verringert die Wahrscheinlichkeit von Bedienungsfehlern.
- **Reduziert das Betriebsrisiko durch zustandsorientierte Orchestrierung**, die Validierung von Voraussetzungen, nachträgliche Bedingungen, automatisches Konfigurationsrendering und automatische Erwartungsvalidierung nutzt.

Die besonderen Vorteile von Root-Cause Identification und Intent-based Analytics

Die betrieblichen Analysekomponenten von Apstra (Ursachenermittlung und Intent-based Analytics) ermöglichen Ihnen Folgendes:

- **Reduzieren Sie die durchschnittliche Reparaturzeit und die Arbeitsauslastung von Fachkräften** durch vereinfachte Betriebsanalysen, sodass Ihre qualifiziertesten Ressourcen ihre Zeit mit Verbesserungen und Innovationen verbringen können, anstatt sich mit Brandbekämpfung zu beschäftigen.
- Weniger Daten erfassen und speichern – **mehr Kenntnisse extrahieren**. Auf der Grundlage des Referenzdesign-Verhaltensstandards weiß Apstra, wonach es sucht, und sammelt nur diese Informationen. Diese Verarbeitung während des Betriebs kann zu einer Verringerung des Speicherbedarfs und der Nachbearbeitung nicht interessanter Daten in einer Größenordnung von fünf bis sechs führen. So können Sie Ihre Infrastruktur effizienter betreiben und bei gleichzeitiger Kostenkontrolle wettbewerbsfähig bleiben.
- **Probleme werden schnell und einfach sichtbar.** Apstra ermittelt wichtige, handlungsrelevante Ereignisse in einer „einfachen Übersicht“ und eliminiert auftretende Symptome, die lediglich Artefakte einer identifizierten Ursache sind.
- **Automatisieren Sie komplexe Workflows.** Das Apstra System ermöglicht es Ihnen, kontextbezogene Workflows zur Fehlerbehebung zu automatisieren (die ansonsten umständlich, ineffektiv, zeitaufwendig und kostspielig sind).
- **Profitieren Sie von vollständig automatisierter Wartung.** Durch ununterbrochene Intent-Synchronisierung ermöglicht Apstra kostenlose Zero-Touch-Wartung bei Veränderungen. Somit ist das System robust, reagiert automatisch auf Änderungen und spart enorme Kosten, die mit der Wartung von Datenverarbeitungs-Pipelines verbunden sind.
- **Vermeiden Sie teure Do-it-yourself-Entwicklungen (DIY)**. Die Eigenentwicklung von integrierten Datenverarbeitungs-Pipelines ist kostspielig und fehleranfällig, kostet Zeit und erfordert Ressourcen, die Sie von Ihrem Kerngeschäft abziehen müssen. Gerade für den Mittelstand bedeutet dies einen sehr hohen Aufwand.
- **Liefern Sie hochpräzise Ergebnisse** im Vergleich zu den Ansätzen, die maschinelles Lernen/künstliche Intelligenz bieten.
- **Setzen Sie auf einen anbieterunabhängigen Ansatz**, der Ihnen die Freiheit gibt, zwischen den besten Anbietern und Funktionen zu wählen.
- **Verwenden Sie gemeinsame APIs** in Ihrer öffentlichen/privaten/Hybrid-Cloud-Infrastruktur.

Skalierbarkeit ab Tag 1: eine Fallstudie

Apstra setzt von Tag 1 auf Skalierbarkeit. Sehr anschaulich wird dies am Beispiel eines Kunden, der damit eine umfangreiche Validierung mit beeindruckenden Ergebnissen durchgeführt hat:

- **Physische Infrastruktur:** 6.000 Validierungen des Schnittstellenstatus; 1.000 Validierungen der Verkabelung; 36.000 Fehlerzählungen; 10.000 Validierungen von Leistungs-, Temperatur- und Spannungsmetriken
- **L2/L2 Data Plane**: 12.000 Warteschlangen-Drop-Zähler; Überprüfung von Hunderten von MLAGs; 3.000 Spanning Tree Protocol (STP)-Überprüfungen; Benachrichtigung über Statusänderungen
- **Steuerungsebene**: 1.500 gesicherte BGP-Sitzungen; ~500 erwartete Next Hops/Standardrouten
- **Kapazitätsplan**: ~500 Trendanalysen mit konfigurierten Schwellenwerten für die Verwendung von Routing-Tabellen; ARP-Tabellen (Address Resolution Protocol); Multicast-Tabellen pro virtuellem Routing und Weiterleitung (VRF); 6.000 Link-Nutzungsüberprüfungen
- **Compliance**: Sicherstellen, dass die erwarteten Betriebssystemversionen auf allen etwa 100 Switches laufen

- **Multicast**: 2.500 Validierungen für erwartete PIM-Nachbarn (Physical Interface Module); 300 Rendezvouspunkt-Überprüfungen; 25 Validierungen hinsichtlich der Erkennung anormaler Muster bei der Anzahl von Quellen, Gruppen, Quellen-Gruppen-Paaren an Rendezvouspunkten

Anstelle einer einzigen Übersicht mit 82.000 Einträgen wurde dem Kunden eine einfache Übersicht präsentiert, die nur Anomalien anzeigte, die nach den Vorgaben des Kunden in einem Dashboard kategorisiert waren:

- Zu 100 % korrekt (und nicht statistisch abgeleitet)
- Keine laufende Wartung erforderlich, da sie ständig mit Topologie und Intent des Kunden synchronisiert waren
- Bereitstellung des relevanten Handlungskontexts (wo gab es Abweichungen, Art der Abweichung, gewünschter Zustand)
- Einsparung von Speicher- und Verarbeitungsbedarf (nur 9 GB RAM, 96 GB Festplatte erforderlich)
- Eliminierung der Gefahr einer internen Bindung und von Altlasten durch das Schreiben komplexer und anfälliger Datenverarbeitungspipelines

Fazit

Die Tatsache, dass Sie nicht in der Lage sind, zuverlässige Änderungen an Ihrer IT-Infrastruktur vorzunehmen, stellt ein großes Hindernis für Wachstum und Innovation dar. Apstra räumt mit dieser Furcht auf und macht es möglich, die gefürchtete „veraltete Infrastruktur“ für immer zu beseitigen, indem es Ihnen die Möglichkeit gibt, Änderungen zuverlässig vorzunehmen, was Ihnen wiederum ermöglicht, zuverlässig Innovationen zu entwickeln und wettbewerbsfähig zu bleiben.

Über Juniper Networks

Juniper Networks vereinfacht mit seinen Produkten, Lösungen und Services die Netzwerke, die unsere Welt umspannen. Durch kontinuierliche Innovation überwinden wir die Einschränkungen und die Komplexität, mit der Netzwerkadministratoren in der Cloud-Ära zu kämpfen haben, und unterstützen unsere Kunden und Partner bei der Bewältigung ihrer größten Herausforderungen. Wir bei Juniper Networks sind überzeugt, dass Netzwerke ein Medium für den weltweiten Wissensaustausch und den die Welt verändernden Fortschritt der Menschheit sind. Deshalb haben wir uns das Ziel gesetzt, bahnbrechende Lösungen für automatisierte, skalierbare und sichere Netzwerke zu entwickeln, die mit dem Tempo unserer schnelllebigen Geschäftswelt Schritt halten.

Unternehmens- und Vertriebshauptsitz
Juniper Networks, Inc.
1133 Innovation Way
Sunnyvale, CA 94089 USA
Telefon: +1 888 586 4737
oder +1-408-745-2000
www.juniper.net

Hauptniederlassung für die Regionen APAC und EMEA
Juniper Networks International B.V.
Boeing Avenue 240
1119 PZ Schiphol-Rijk
Amsterdam, Niederlande
Telefon: +31-0-207-125-700

JUNIPER NETWORKS | Engineering Simplicity