Network Policy Control: Methods and Considerations

An Industry Whitepaper

Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Executive Summary</td>
<td>1</td>
</tr>
<tr>
<td>Introduction to Policy Control</td>
<td>2</td>
</tr>
<tr>
<td>Functional Components</td>
<td>2</td>
</tr>
<tr>
<td>Identifying Convergent Conditions</td>
<td>3</td>
</tr>
<tr>
<td>Evaluating Business Logic</td>
<td>3</td>
</tr>
<tr>
<td>Enforcing Policy Actions</td>
<td>3</td>
</tr>
<tr>
<td>Types of Policy Elements</td>
<td>3</td>
</tr>
<tr>
<td>Methods and Considerations</td>
<td>4</td>
</tr>
<tr>
<td>Policy Expression</td>
<td>4</td>
</tr>
<tr>
<td>Identification: Available Conditions</td>
<td>4</td>
</tr>
<tr>
<td>Evaluation: Linking Conditions to Actions</td>
<td>5</td>
</tr>
<tr>
<td>Enforcement: Taking Action</td>
<td>8</td>
</tr>
<tr>
<td>Example: Congestion Management</td>
<td>9</td>
</tr>
<tr>
<td>Access Type Dependencies</td>
<td>10</td>
</tr>
<tr>
<td>Control and Data Plane Synergy</td>
<td>11</td>
</tr>
<tr>
<td>Policy Decision Points</td>
<td>12</td>
</tr>
<tr>
<td>Other Considerations</td>
<td>14</td>
</tr>
<tr>
<td>Usability</td>
<td>14</td>
</tr>
<tr>
<td>Extensibility</td>
<td>14</td>
</tr>
<tr>
<td>Scalability</td>
<td>15</td>
</tr>
<tr>
<td>Conclusions</td>
<td>16</td>
</tr>
<tr>
<td>Related Resources</td>
<td>16</td>
</tr>
<tr>
<td>Invitation to Provide Feedback</td>
<td>16</td>
</tr>
</tbody>
</table>

Executive Summary

Network policy control is an indispensable technology for ensuring the efficient and effective delivery of Internet data services. Communications service providers (CSPs) use network policy control to enable a wide variety of solutions across a broad range of functions. With so many factors in play, CSPs have much to consider when determining the best approach to policy management for their network.

This paper explores how CSPs can benefit from the convergence of compatible features in supporting desired use cases with minimal overhead.
Introduction to Policy Control

Network policy control is an indispensable technology for ensuring the efficient and effective delivery of Internet data services. Communications service providers (CSPs) use network policy control to enable a wide variety of solutions across a broad range of functions, including:

- to gain valuable business intelligence for informed decision-making
- to create new revenue streams by powering new services
- to optimize data delivery for improved efficiency and higher subscriber quality of experience
- to protect subscribers and the network from security threats

Network policy control, and the related (but slightly narrower) concept of policy management, is fundamentally about achieving business objectives by specifying how the network itself behaves in response to an infinitely diverse stream of events and conditions across the data, control, and business/operations support systems (B/OSS) planes.

More succinctly: network policy control provides the network with the decision-making intelligence and enforcement capabilities to bring business objectives to life. Table 1 shows some straightforward business objectives, and, in plain language, interprets these objectives as instructions for a policy control solution.

<table>
<thead>
<tr>
<th>Business Objective</th>
<th>Policy Control ‘Instructions’</th>
</tr>
</thead>
<tbody>
<tr>
<td>Understand the end user quality of</td>
<td>For each application type (e.g., video, web browsing, voice-over-IP, gaming, etc.) and overall (i.e., for all traffic), quantitatively measure and clearly present quality of experience scores.</td>
</tr>
<tr>
<td>experience</td>
<td></td>
</tr>
<tr>
<td>Prevent subscriber bill shock</td>
<td>Measure per-subscriber usage in real time, then trigger a notification (e.g., via SMS, in-browser, etc.) at appropriate thresholds (e.g., 80%, 95%, 100%)</td>
</tr>
<tr>
<td>Implement shared data plans</td>
<td>Concurrently measure usage for each user and aggregate the total, while comparing to available family or multi-device quota</td>
</tr>
<tr>
<td>Prevent network congestion</td>
<td>Detect network congestion in real time, and trigger precise management to protect high-value applications for the maximum number of subscribers</td>
</tr>
<tr>
<td>Mitigate DDoS attacks</td>
<td>Detect, automatically mitigate, and report on Distributed Denial of Service (DDoS) attacks.</td>
</tr>
</tbody>
</table>

Functional Components

A network policy is the technological implementation of a CSP’s desired business objectives for their network. There are three distinct functions in any network management policy:

1. Identify **convergent conditions/events** in the network
2. Evaluate **business logic** to determine the appropriate response behavior
3. Enforce **policy actions** to achieve the desired result

These functions are briefly introduced in the next three subsections, and are subsequently explored in greater detail later in this document.
Identifying Convergent Conditions
There are many conditions occurring within the network (and its supporting systems) that CSPs can incorporate into their overall policy management design in support of a particular business objective.

Condition examples include:

- The classification of data plane (i.e., user traffic) protocol and flow characteristics; in other words, what the traffic is
- The detection of data plane events (i.e., something specific happens) and measurement of its characteristics (e.g., values including volume, counts, duration, etc. and characteristics like resolution, codec, device, browser, user location, etc.)
- Specific control plane events (e.g., a subscriber connects to the network) and information provided by B/OSS elements (e.g., that subscriber’s plan details)

Evaluating Business Logic
Bridging the gap between conditions-as-inputs and actions-as-outputs is the decision-making logic that translates business objectives into network management policies.

A solution’s ability to evaluate ‘what to do’ or ‘how to handle’ the rich diversity of control and data plane conditions and events is largely determined by how it expresses policy. That is, how effective a solution is at turning abstract (and, oftentimes, deceptively simple) business objectives and concepts, which are expressed in human language, into instructions for network systems.

Enforcing Policy Actions
Once conditions and events have been evaluated by a policy, actions enforce the desired outcome in support of the overall business objective.

An action might impact how traffic flows through the network (e.g., traffic steering/redirection, prioritization, etc.), or an action might only exist in the control or B/OSS plane (e.g., as a statistic being written to the database, or decrementing a quota allowance, or noting a change of state).

Types of Policy Elements
At the very highest level, policy elements are hardware and software platforms that host expressed policies and perform one or more of the three functions (e.g., identify, evaluate, enforce) to transform a CSP’s ideas into the reality of desired network operation.

These elements can be components of the network transport infrastructure, or an industry-defined, standards-based network element, or a vendor product that fulfills a particular use case. Examples include:

- PCRF (Policy and Charging Rules Function)
- PCEF (Policy Charging and Enforcement Function)
- TDF (Traffic Detection Function)
- OCS (Online Charging System) and OFCS (Offline Charging System)
- CMTS (Cable Modem Termination System)
- P-GW/PDN-GW (Packet Data Network Gateway)
- Specialized solutions that evaluate and/or enforce network policies (stateful firewalls, deep-packet inspection appliances, cloud services policy controllers, transparent caches, parental control filters, etc.)
Network Policy Control: Methods and Considerations

Methods and Considerations

There are a number of important criteria to consider as part of an overall policy management design or when evaluating potential solutions. The sections and subsections that follow relate each criterion to the identification, evaluation, and/or enforcement components of the network policy control solution.

Policy Expression

The method of policy expression largely determines how a solution evaluates sets of conditions and what it can do in response. This is the real-time ‘thinking’ component of policy management, where human ideas about what to do in a particular situation are expressed in technical terms to achieve the desired outcome. For example, consider Table 2, which shows how the examples introduced in Table 1 can be achieved in terms of linking conditions to actions.

Table 2 - Relating business objectives with conditions and actions (conditions in bold, actions in italics)

<table>
<thead>
<tr>
<th>Business Objective</th>
<th>Conditions and Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Understand the end user quality of experience</td>
<td>Measure network latency and measure video buffering(^1) and measure video bailouts and measure CDN performance and publish measurements to database</td>
</tr>
<tr>
<td>Prevent subscriber bill shock</td>
<td>If subscriber usage is 80% then send notification</td>
</tr>
<tr>
<td></td>
<td>If subscriber usage is 100% then redirect browsing to a captive portal</td>
</tr>
<tr>
<td>Implement shared data plans</td>
<td>If subscriber Jeff quota used 50 MB then deduct from Smith family plan</td>
</tr>
<tr>
<td></td>
<td>If subscriber Ann quota used 5 MB then deduct from Smith family plan</td>
</tr>
<tr>
<td>Prevent network congestion</td>
<td>If resource is congested and subscriber is on resource and subscriber is recent heavy user and protocol is software update then shape and log enforcement action</td>
</tr>
<tr>
<td>Mitigate DDoS attacks</td>
<td>If traffic behavior matches SYN flood profile then block and log instance and notify operator</td>
</tr>
</tbody>
</table>

Identification: Available Conditions

The set of actual conditions available to the policy control solution is arguably the largest determinant in the solution’s overall utility.

For instance, each new condition (and, taking it even farther, the ability to take distinct actions based on distinct values of the condition) increases, exponentially, the diversity of situations that can be considered. In this respect, conditions are even more valuable than actions, as the total list of available actions is somewhat finite.

It can be argued that the more a solution can evaluate multiple conditions in support of many possible actions, the better it can deliver value in support of rich use cases that achieve specific business goals. For the mathematically inclined:

\[
\text{Solution Value} \approx (\text{Number of Conditions})^2 \times (\text{Number of Actions})
\]

\(^1\) Buffering applies to progressive video streaming; for adaptive video streaming, one would measure upshifts and downshifts. Plus, this example oversimplifies measuring video quality of experience, which is explained in detail in Measuring Internet Video Quality of Experience from the Viewer’s Perspective.
A condition can be as simple as the application protocol in use, and as rich as a complex behavior that must be carefully measured over time (e.g., does a user exhibit behavior indicative of a malware infection?).

Conditions\(^2\) must first be identified before anything can be done about them, and they can come from many different sources. In the simplest separation, a condition can be ‘created’ by a device (e.g., a TDF identifies a packet as being a particular application) or can be supplied by a device (e.g., a policy controller receives channel utilization information from a CMTS).

The origin of the condition doesn’t really matter - so long as a system has access to as many conditions as possible, then great! - but there is enormous variation in the number of conditions available to different systems. Some systems can directly measure (i.e., create) many conditions, some can only measure a few; some systems possess the interfaces required to connect to, and receive conditions from, many different network elements, while others operate in relative isolation.

**Conditions on the Data Plane**

Data plane conditions deal mainly with understanding the nature of subscriber traffic\(^3\):

- What type of traffic is it (application protocol)?
- What is it doing (streaming a video, delivering web content, conducting a DDoS attack, etc.)?
- What are its characteristics (e.g., volume, number of events, duration, quality of experience, bitrate, codec, round-trip time latency, etc.)?
- Is this network resource congested right now?
- What is the total amount of video traffic being carried by the network right now?

**Conditions on the Control Plane**

The control plane enables the network to function (e.g., allowing subscribers to log-on to the network, permitting hand-offs between mobile towers, exchanging usage information between billing systems, etc.); conditions on the control plane tend to revolve around the meta-information that is associated with those enablement functions:

- A subscriber logs on to the network: who is it?
- To what mobile tower did the subscriber connect?
- What is the quality of service requirement (QoS Class Identifier) for the application in use?
- What are the subscriber’s entitlements (e.g., OCS updating a PCRF on available data quota, what service plan does the subscriber have, etc.)?
- A Diameter Gx policy enforcement instruction is received

**Evaluation: Linking Conditions to Actions**

The evaluation function examines one or more conditions and outputs one or more actions, to achieve some goal.

This simple characterization betrays immense complexity behind the scenes, and variation between solution capabilities. At the highest level, there are three areas that must be examined:

---

\(^2\) This document somewhat interchanges “condition” and “event”, hopefully without any loss of clarity. Strictly speaking, a condition requires a comparator, so one could say that a condition, as it relates to an event, is that the event went from a state of not existing to a state of existing: expressed differently, “if event = exists, then do something”.

\(^3\) For a complete overview of the issues surrounding Internet traffic classification, see Identifying and Measuring Internet Traffic: Techniques and Considerations.
Network Policy Control: Methods and Considerations

- Examining conditions
- Rules architecture
- Triggering actions

In the ideal implementation, the solution can implement precisely what the operator wants (e.g., congestion management that only acts when and where congestion manifests, and impacts the fewest number of subscribers); in a less-optimal scenario, the operator has to rethink what they want in order to match the limitations of the solution (e.g., network-wide traffic shaping in the evening).

Understanding these three areas will let a CSP determine with what type of system with which they are working.

**Examining Conditions**

While the previous section explained what different events and conditions are, this section deals with what can be done with them.

For any given event, policy evaluation lets a system check a condition, but the checks available vary from the simple (e.g., is X true, or false?) to the more complex (e.g., does X have a value that exists in this known set?).

Communications service providers should ask if a policy control system supports:

- True/false assessment
- Numerical values and comparators (e.g., greater than, less than, equal to)
- Percentage values and comparators (e.g., is X greater than 50% of Y?)
- Set membership
- String functions

Things get particularly powerful when more than one condition is examined concurrently. For instance, the statement *if the traffic is video streaming and it’s going to an iPhone* looks for two conditions to be simultaneously satisfied: a traffic type, and a device type.

CSPs should ask vendors to explain multi-condition capabilities in detail, for two reasons: first, capabilities in general vary enormously; second, system architectures can have important ramifications for this area, as we will see shortly.

**Rules Architecture**

The evaluation of conditions determines what actions a solution will take. Taken together, the complete statement “if (these conditions are met) then (take these actions)” is most frequently described as a rule. A collection of rules that are all related is often referred to as a policy (e.g., “our bill-shock notification policy is composed of a handful of rules.”)

The actual manner in which a system evaluates rules can have enormous impacts on its overall capabilities and usability.

Many people who evaluate policy control solutions are familiar with how firewalls work, and the most simple policy control devices behave in a similar manner: there is an ordered hierarchy of rules, and for each event the system sees it starts at the top of the rules list and proceeds to each subsequent rule until the conditions are satisfied and the rule is executed; upon exit, the system does not go any deeper into the rules list.
This model is simple and straightforward, but suffers from a few significant limitations: first, there is often a hard limit on the number of rules; second, the operator must carefully construct a linear hierarchy to represent what is probably a much more complicated and nuanced set of business objectives; third, if business priorities or objectives change, or new use cases are sought, then that strict prioritized list of rules might need to be completely rebuilt; fourth, to maximize performance the most frequently-triggered rules should be placed high up in the list.

In short, the real world typically isn’t linearly prioritized. This type of system, while easy to get up and running, quickly forces the operator to make mental contortions to fit a strict architecture.

A related problem is that of exponential rule explosion. To illustrate this issue, consider a system that can consider many conditions at once, but for simplicity’s sake let’s say that the conditions are only evaluated on a true or false basis.

Some policy control systems work by dividing traffic between virtual tubes, each of which gets some amount of reserved bandwidth. Each data packet is examined to determine to which virtual tube it belongs, and each packet can only be assigned to a single virtual tube. Each virtual tube has a corresponding rule in a linear hierarchy, and the rules consider many conditions in order to provide the CSP with very granular control.

So far, things look very promising, but there is a critical problem: each new condition that is considered doubles the number of rules, meaning that the total available rules and tubes is quickly exhausted. A single binary condition places a packet in one of two virtual tubes; linking two binary conditions together gives more precise control, but requires four tubes; three conditions need eight tubes; four conditions need 16 tubes.

Today’s business use cases take into account many different factors - traffic type, subscriber plan, subscriber location, device, presence or absence of network congestion, recent usage history, and so on - so running out of rules or virtual tubes is a very real problem.

Often, the operator is faced with three options: purchase another policy element and deploy in serial; sacrifice some existing use cases for the desired new one; or abandon the new use case.

At the leading edge of policy control technologies are ‘freeform’ systems that can identify an entire set of applicable rules (i.e., as opposed to finding a first match and then exiting the search), each of which triggers separate actions. These systems, by their very nature, offer enormous benefits for network operators:

- They allow much greater flexibility in expression of business objectives, because conditions need not be artificially linked together; instead, different policies (each consisting of a collection of rules) can be expressed independently but are still guaranteed to be evaluated. This characteristic has important benefits for ongoing maintenance and extension: new policies can be added, independently, at any time, without having to re-examine the existing collection of policies.
- Because they are not consuming from a finite pool of deterministic outcomes, there is no concept of a finite rules limit

Triggering Actions
To provide some practical illustration of the concepts that have just been presented, consider three separate business objectives, each expressed as a policy:
• When a user hits 80% of his or her quota, provide a notification
• When the network is congested, prioritize video streaming traffic
• If the device in use is an Amazon Kindle, do not count it towards the user’s quota

With a rigid, linear system, to behave correctly when a video packet destined for a Kindle happens to push consumption past 80%, those policies would need to be translated into a monstrous list of rules ($2^x$ rules, if we assume $x$ binary conditions) to account for all possible condition values, and an associated $2^x$ of the finite supply of virtual tubes. The packet is considered against each rule, in turn, until all conditions are matched, and then the packet enters the corresponding virtual tube.

With a freeform system, each packet that enters the system is simultaneously examined against all three of these policies. Some packets will match on one, some on two, and some on all three. The freeform system keeps track of which policies should be applied, and then applies the combination of policy actions required.

**Enforcement: Taking Action**
Actions are the final result stemming from the evaluation of conditions as defined by the policy; they include, but are not limited to, the options shown in Table 3.

<table>
<thead>
<tr>
<th>Enforcement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>QoS Mark</td>
<td>Use DSCP to mark packet for downstream/upstream prioritization</td>
</tr>
<tr>
<td>Record/Report/Log</td>
<td>Record conditions, measurements, and actions for reporting and auditing purposes</td>
</tr>
<tr>
<td>Signal</td>
<td>Send standards-compliant signal to another element with instructions</td>
</tr>
<tr>
<td>Shape/Throttle</td>
<td>Directly shape data plane throughput</td>
</tr>
<tr>
<td>Block</td>
<td>Block traffic (for instance, if the packets are part of an attack)</td>
</tr>
<tr>
<td>Redirect</td>
<td>Intercede to direct subscriber traffic to a service function chain, or a walled garden/captive portal</td>
</tr>
<tr>
<td>Meter</td>
<td>Record data usage for charging purposes, or in support of fair use policies</td>
</tr>
</tbody>
</table>

There are various methods of enforcement and places on the network where enforcement can occur:

• A policy control solution could enforce policies locally, for instance as traffic passes through the system; or
• It could trigger remote enforcement, perhaps by initiating a Diameter Gx or Rx message that is acted upon by another system; or
• The system could be a hybrid: the policy control solution directly changes a DSCP mark on a packet, and that mark is subsequently acted upon by another system

Figure 1 illustrates these three options.
Network Policy Control: Methods and Considerations

Example: Congestion Management
Consider the common business (and technical) objective of congestion management. An access network resource experiences congestive collapse when it hits maximum throughput capacity. Associated with this congestive collapse is an exponential increase in latency and a plummeting subscriber quality of experience.

Network policy control provides various methods of managing congestion, the ultimate goal of which is to maximize the length of time each access resource can deliver good subscriber QoE while impacting the fewest subscribers possible with management actions. 4

An effective congestion management solution, at the highest level, has to do two things:

1. Detecting congestion
2. Managing congestion

Congestion detection is based on a solution’s ability to evaluate conditions.

A solution might use time-of-day as the sole criterion for determining (in this case, guessing) when the network is congested. Obviously, time of day is not ideal since the network could be congested outside the specified time frame, and might not be congested during the specified time frame. An approach based on a time-of-day condition would therefore fail to satisfactorily achieve our ultimate goal.

Plus, we haven’t yet considered the network’s resource hierarchy/topology. In practice, congestion occurs on individual resources, rather than in the network as a whole, so this means we should have knowledge of network topology as a required condition.

Extending our example, a more sophisticated evaluation could be based on the number of subscribers concurrently using a shared network resource as an indirect indicator of congestion. This approach

---

4 For more information see [Network Congestion Management: Considerations and Techniques](#)
combines two conditions: access resource identity and a count of concurrent subscribers. No doubt it will be better at achieving our goal than the time-of-day method, but can we do better?

An even more accurate approach to congestion detection is to combine access resource identity with the dynamic measurement of an actual indicator of congestion detection: packet latency.

By combining a plurality of conditions, and by choosing the right ones to use in that combination, we can precisely respond to an imminent congestive collapse.

Now, let’s turn our attention towards congestion mitigation/management. Further, let’s assume that we have chosen the final congestion detection mechanism outlined above, so we have real-time knowledge of specific resources that are experiencing congestion.

We have several actions available to mitigate the congestion. For instance, when congestion is detected, a solution can manage a specific application category of traffic, such as Peer-to-Peer Filesharing, on the congested resource. Alternatively, if the solution could incorporate the appropriate conditions, it could de-prioritize all or a specific subset of application traffic for those users who have consumed the most data in the past 15 minutes (and again, only on the congested resources).

Clearly, CSPs benefit most when they are free to combine multiple conditions and multiple actions into one policy ‘narrative’ that achieves the desired effect while avoiding unwanted consequences.

**Access Type Dependencies**

Network policies are realized by communications and signaling that occur on the data and control plane using a wide variety of methods (e.g., data plane flow and subscriber characteristics, Diameter signaling, PCMM signaling, Radius, LDAP, DHCP, REST/SOAP, IPDR, etc.).

The heterogeneity of the policy management spectrum is a challenge CSPs must constantly overcome in the pursuit of uniform/consistent implementation of business objectives.5

To illustrate this complexity, Table 4 shows a sample of the different standards involved in implementing business objectives on different network types.

<table>
<thead>
<tr>
<th>Defined Policy</th>
<th>Network</th>
<th>Standard/Protocol/Interface</th>
<th>Activities</th>
</tr>
</thead>
<tbody>
<tr>
<td>If subscriber usage is 80% then send notification</td>
<td>3G/3GPP</td>
<td>Radius</td>
<td>Metering</td>
</tr>
<tr>
<td>If subscriber usage is 100% then captive portal</td>
<td></td>
<td>Diameter Gx</td>
<td>In-browser notification</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Diameter Gy</td>
<td>Redirect to captive portal</td>
</tr>
<tr>
<td>If subscriber John quota used 50 MB then deduct from Smith family plan</td>
<td>DOCSIS Cable</td>
<td>PCMM/COPS, IPDR, DHCP, DPI-based Traffic Classification</td>
<td>Measure usage via IPDR, Detect protocol for app-specific plans, Update billing system</td>
</tr>
<tr>
<td>If subscriber Rita quota used 100 MB then deduct from Smith family plan</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

5 Why is uniformity of application important? A CSP probably doesn’t want to have to explain to subscribers that the video optimization service, or the zero-rated music streaming promotion, is only available on parts of the network built by Vendor A, because Vendor B doesn’t support the same capabilities or the required interfaces.
Network Policy Control: Methods and Considerations

<table>
<thead>
<tr>
<th>Condition</th>
<th>Access Type</th>
<th>LDAP</th>
<th>In real-time</th>
</tr>
</thead>
<tbody>
<tr>
<td>If resource is congested and subscriber is on resource and subscriber is recent heavy user and protocol is software update then shape and record enforcement action</td>
<td>DSL</td>
<td>Radius, SOAP, Vendor-proprietary DPI, policy decision and enforcement</td>
<td>Measure latency, Detect congestion, Measure usage, Detect protocol, Shape</td>
</tr>
<tr>
<td>Enable VoLTE Call</td>
<td>LTE</td>
<td>QCI, Diameter Rx, SIP</td>
<td>Signal VoLTE call, Establish bearers, Maintain minimum latency standard</td>
</tr>
</tbody>
</table>

Today, CSPs increasingly find themselves managing policy for many different access types merged into one network. A key question is how to manage a core set of policies that can apply to all of them at once while maintaining the required compliance standards defined by various network industries.

An ideal policy control solution has no access type dependencies and will apply universally across the whole network; that is, the network operator can define their business objectives, and they will be implemented consistently and correctly across the entire network, using the appropriate standards, regardless of the particular access technology on which a subscriber's traffic is being carried at any instant.

**Control and Data Plane Synergy**

Policy management enables use cases for both the control plane and the data plane. For example, a PCRF has the capability of understanding the service entitlements of individual subscribers, receiving usage information over Gx, and signaling instructions to an enforcement element over Gx. The PCEF/TDF can understand the application protocol per flow, meter data and take action to manage data plane traffic.

When these two elements - the PCRF and the PCEF - need to work together, the available use cases can dwindle based on vendor interoperability, the intersection (or not) of compatible features, and the complexity of the network (e.g., a converged cable/mobile network can have two completely different sets of decision and enforcement elements).

To visualize this problem abstractly, a PCRF could have a set of features and, correspondingly, supported use cases \{a, b, c, d, e, f\} and a PCEF/TDF might support its own set of features and use cases \{e, f, g, h, i\}. Independently, this seems like a full spectrum of functionality (e.g., a through i), but when the two elements are forced to work together in a standards-compliant environment the overall solution could be limited to just supporting the overlapping features e and f, and nothing else.

The ability to extend a single policy management framework across both planes is therefore an important consideration for CSPs. At the highest level, there are two options:

1. Exhaustively ensure that both solutions support compatible feature-sets and can both enable the desired use cases
2. Source a solution that spans both the control plane and data plane
There are pros and cons to either approach.

To demonstrate how policies in both the control and data plane can work together to enable innovative service offerings, Figure 2 shows an example of family plan processing where the PCRF is performing Gx Usage Monitoring.

![Figure 2 - Specific Family Plan quota management and billing example](image)

The PCEF counts usage and associates it accurately with the correct subscriber, while the PCRF manages a device-specific quota plan for each family member. This enhanced use case is only possible when both the PCRF and PCEF elements can work seamlessly in tandem.

**Policy Decision Points**

Policy Decision Points (PDPs) are the places in the network where expressed policies conduct their evaluations (i.e., decide what to do about a particular situation).

In most reference architectures, the data plane elements are considered to be relatively dumb devices that provide information about observed traffic characteristics to a central authority and/or enforce policies handed out by that central authority. The central authority makes all the decisions.

![Figure 3: Generalized Centralized Architecture](image)
While this approach maintains a very ‘clean’ functional separation, that separation can have unintended consequences.

**Latency, Signaling Storms, and Distributed Decisions**

When the result of a decision is an action that affects network traffic, latency is an important factor to consider. There is a latency cost each time a policy management implementation breaches the data/control plane barrier, such as the PCEF asking the PCRF what to do about each flow and packet.

This latency has the potential to adversely impact subscribers. For instance, consider a URL filtering policy that requires every URL request be ‘cleared’ by the PCRF. In this scenario, the subscriber’s request must be held up until the decision to allow or block content is returned. In effect, a CSP simply trying to adhere to government regulations would be negatively impacting all web browsing.

The ability to make decisions, or evaluate, policy within the data plane reduces latency by keeping the evaluation process close to the enforcement point, and results in significant performance benefits.\(^6\)

---

\(^6\) For a complete overview of the advantage of distributed policy decisions, see the whitepaper *Distributed Decisions in Policy Control.*

---

**Figure 4: Generalized Distributed Architecture**

A second argument in favour of embedding (at least some) intelligence in the data plane is the amount of signaling required under the strictest (and most common) interpretation of the reference architectures, in which the PCEF asks the PCRF what to do for each and every flow. In fact, an entire industry has sprung up to provide Diameter routing agents that exist solely to manage this signaling storm. Surely this was not the intent, but it is the reality.

By making decisions in the data plane element wherever possible, CSPs can reduce the amount of control signaling by several orders of magnitude. Consequently, the control plane can be scaled down, signaling load balancers are no longer needed, and the network becomes easier to operate.
Other Considerations

Usability
Usability is a complex issue because it is impacted by user type and solution adaptability factors that are difficult to quantify. Therefore, it is best for a CSP to ask a range of questions and then weigh the responses based on personal preference and priority.

Getting Up and Running
No doubt a CSP has some ideas about what use cases are most important, and they want to get up and running quickly, but the extent to which this is possible varies based upon the usability of the system.

A simple system might allow a CSP to quickly deploy policies right away; a more complex system might take longer, but could deliver a better result. Complexity can also be mitigated by using tools like policy templates for common use cases.

User Types and Interfaces
CSPs find it useful when a solution offers functionality and views tailored to the capabilities of specific user types based on their scope of knowledge and assigned tasks.

For example, a junior operator might have a few settings they would want to adjust and tweak, whereas a senior network engineer might want to define a complex policy from scratch.

It is therefore helpful for a CSP to ask about the range of user interfaces that are available, and the level of expertise required to use each. For instance, is there an interface that allows the junior operator to modify values and variables without ‘touching’ the underlying foundational policies? Is there an interface that allows a super-user complete access? Is there a GUI, or is all policy expression performed within an IDE-like environment?

Extensibility
Policies are always evolving: business priorities change, regulations are drafted, standards evolve, use cases are created, acquisitions lead to integration with new networks and technologies, new Internet applications emerge, etc.

The degree to which a CSP can take their existing policy deployment and extend it in new directions will play a large part in their overall success in the marketplace.

Some approaches offer the ability to change only a few settings for an established core policy that can never be altered in the field. Other approaches provide full access to policy creation and adaptation in the field, but it requires a significant amount of learning for the user.

Vendors who build their businesses around service revenue are happy to build a custom solution - for a large investment and with an 18-month delivery timeline.

Sometimes the easy-out-of-the-box solution shows limitations once a CSP tries to move beyond a small handful of simple use cases.

In short: there is enormous variation from vendor to vendor, and CSPs must thoroughly investigate extensibility during the evaluation phase, to avoid being severely limited once a solution has been rolled out into production.
Scalability
Scalability is another immensely complicated area, but it is useful to consider it first from a few angles: CPU/processing requirements, supported rule-space, and stateful awareness.

Processing Capacity
Every action a policy control solution takes will consume some processing capacity, and it is important that a CSP be able to understand the relationships between policies and processor utilization (for instance, “If I turn this on, what will happen to my CPU load?”). In general, dedicated policy control devices have far more available processing than embedded devices, but within the realm of dedicated policy control devices the distinction is less clear.

Policy Rules
Another important consideration is the number of policy ‘rules’ that a system supports. Many systems are based on a simple approach that has a finite capacity for rules (usually the number of supported rules is a power of 2, because they are limited to combining binary conditions). With such systems, a CSP can quickly exhaust or run out of rules; once that point is reached, each new thing a CSP wants to achieve requires eliminating something that was already being done.

Even a system that supports hundreds of thousands of rules can quickly exhaust the rule-space by linking conditions together. In such a system, each linked condition (i.e., considering \( A \) and \( B \) at the same time) exponentially increases the number of rules occupied by the use case: a use case dependent upon one condition, \( A \), uses two rules (one is executed if \( A \), and one if \(!A\)\); a use case dependent upon two conditions uses \(2^2 = 4\) rules (if \( A \) and \( B \), if \( A \) and \(!B\)\), if \(!A\) and \( B \), if \(!A\) and \(!B\)\); a use case dependent upon three conditions uses \(2^3 = 8\) rules, and so on. Like the famous (and likely apocryphal) story about grains of wheat on a chessboard, things multiply up very quickly.\(^7\)

Other systems have no such limitation - there is literally no hard limit on the number of rules. Instead, they are limited by memory required to store the rules themselves, and that’s a very different concept entirely. Such systems will practically support millions of rules, and in fact the very terminology of an individual ‘rule’ is likely misplaced.

Stateful Awareness
Another limiting factor faced by all stateful systems is the memory devoted to maintaining that state. Each flow or subscriber for which a system is keeping state will consume some amount of memory, and as a consequence each system has a dimensioning guideline around the maximum number of concurrent flows or concurrent subscribers. While these values may be tweaked/configured, they are drawing from the same finite resource: memory.

---

\(^7\) See the origin of the story here: https://en.wikipedia.org/wiki/Wheat_and_chessboard_problem#Origin_and_story
Conclusions

With so many factors in play, CSPs have much to consider when determining the best approach to policy management for their network. Network Policy Control can benefit from the convergence of compatible features in supporting desired use cases with minimal overhead.

<table>
<thead>
<tr>
<th>Policy Consideration</th>
<th>Questions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expression</td>
<td>Is the policy expression method compatible with a CSP’s strategic and tactical business goals?</td>
</tr>
<tr>
<td>Flexibility</td>
<td>How flexible is the solution at realizing the exact business goals CSPs conceptualize?</td>
</tr>
<tr>
<td>Extensibility</td>
<td>How is a policy management solution affected by time? Do hardware changes affect policy? Will/can the solution keep pace with evolving standards?</td>
</tr>
<tr>
<td>Universality</td>
<td>To what extent can the policy management solution address network heterogeneity to maximize efficiency and capitalize on network-specific and cross-plane business goals/use cases.</td>
</tr>
<tr>
<td>Scalability</td>
<td>Does the solution grow at pace with all other growth factors?</td>
</tr>
<tr>
<td>Usability</td>
<td>Is solution functionality segmented and contextualized per use type? To what extent can a user affect change?</td>
</tr>
</tbody>
</table>

Related Resources

For more information please see Technology Showcase - the Policy Engine and SandScript.

Invitation to Provide Feedback

Thank you for taking the time to read this whitepaper. We hope that you found it useful, and that it contributed to a greater understanding of network policy control.

If you have any feedback at all, then please get in touch with us at whitepapers@sandvine.com.