Executive Summary

To be a viable solution in modern communications networks, stateful packet-processing equipment must overcome the challenges posed by routing asymmetry and immense scale.

Routing asymmetry increases network efficiency and redundancy, but also presents serious challenges that must be overcome so that CSPs can implement critical use cases such as accurate charging, policy-based measurements, and congestion management.

The rapidly increasing volume of Internet traffic worldwide also presents issues of efficiency, extensibility, and redundancy.

A cluster of Policy Traffic Switch units emulates the behavior of a single enormous network policy control box in order to overcome asymmetry, deliver extreme scale, and preserve redundancy; importantly, clustering achieves these goals with incredible efficiency. The PTS architecture (at a per unit level and at a cluster level) is ‘shared nothing’, which ensures linear extensibility and enables economical N:N+1 redundancy.

Importantly, a PTS cluster is managed as a single unit, so there is no added operational management complexity, and does not introduce any meaningful latency.

Critically, PTS clustering overcomes asymmetry and delivers scale in any access network, with any combination of access technologies, and completely independent of the network’s routing.

PTS clustering is a beautiful, elegant, and - above all - effective and efficient solution to the challenges imposed by routing asymmetry and the demands of massive scale.
Introduction to Scaling Challenges for Stateful Solutions

To be viable solution platforms in modern communications networks, stateful packet-processing equipment must overcome the challenges posed by widespread routing asymmetry and immense scale.

Routing Asymmetry

Traffic asymmetry is widespread and pervasive throughout networks, and broadly speaking takes two forms: Flow Asymmetry, occurring when a flow’s packets traverse different links; and IP Asymmetry, occurring where multiple flows from the same IP address traverse different links.

Routing asymmetry increases network efficiency and redundancy, but also complicates some other matters.

In the context of network policy control, routing asymmetry causes serious challenges that must be overcome so that CSPs can implement critical use cases such as accurate charging, policy-based measurements, and congestion management.

To be a viable policy control platform, a system deployed in the network’s data path must be able to operate in the presence of all types of asymmetry, for all types of traffic. If these conditions aren’t met, then the CSP will not be able to make full, consistent use of policy control.

There are a handful of approaches to overcoming asymmetry, but only three actually preserve complete functionality: deploying only where traffic is guaranteed to be symmetric, intersecting all possible paths with one (large) platform, and clustering to preserve processor core affinity.

Of these three approaches, only clustering offers a combination of versatility, efficiency, and an attractive redundancy model.¹

Absolute Scale

With Internet traffic volumes growing quickly worldwide, on all access technologies, as the combined results of more users and more volume per user, any equipment that intersects this traffic must be able to cope with demands of scale.

While the challenges of scale also apply to individual units, for stateful solutions the primary challenge is of the scale of the entire deployment as a whole. That is, the primary challenge isn’t whether or not a single unit can intersect (and inspect, and apply policy to) a particular volume of traffic; rather, the challenge is whether or not all the units together can intersect and keep state for all the links that are intersected.

Practically, this is the difference between a unit intersecting a 50-60 Gbps (an impressive technological feat, nonetheless) and many units collectively intersecting and maintaining state over many hundreds of gigabits per second or even terabits per second.

With issues of scale come the inevitable (and closely linked) issues of efficiency, extensibility, and redundancy.

¹ More information about the subject of routing asymmetry and its implications for stateful network policy control is available in the Sandvine whitepaper Applying Network Policy Control to Asymmetric Traffic: Considerations and Solutions
Sandvine’s Policy Traffic Switch Clusters

The Policy Traffic Switch (PTS) is Sandvine’s PCEF/TDF appliance, and embeds the Sandvine Policy Engine in the data plane of any access network, with any combination of access technologies.\(^2\)

On a per-unit level, the PTS is impressive, but perhaps the most differentiating characteristic is the capability to combine multiple PTS units into a cluster that behaves as a single enormous platform.

Clustering overcomes the practical limitations of the “one giant box” approach by emulating the behavior of that single box across many in order to overcome asymmetry, deliver extreme scale, and preserve redundancy; importantly, clustering achieves these goals with incredible efficiency.

In short, clustering works by combining the intersection and processing capacity of many units into one and ensuring that all packets associated with a flow, session\(^3\), and subscriber are always processed by the same processor core, regardless of the interface by which they enter the cluster.

Importantly, a PTS cluster is also managed as a single unit, so there is no added operational management complexity. To further simplify life for the operations team, the cluster is self-configuring and self-managing. Once a PTS unit is assigned to a cluster, discovery protocol packets are sent out over the cluster links so PTS units become aware of each other. PTS units are dynamically and automatically added to, and removed from, a cluster through these discovery packets. Once a PTS is a member of a cluster, it sends out keep-alive packets. This process allows each PTS within the cluster to be aware of all the other PTSS in the cluster and to distribute traffic equally.

Clustering is a beautiful, elegant, and - above all - effective and efficient solution to the challenges imposed by routing asymmetry and the demands of massive scale. The sections that follow explain the behavior of clustering in greater detail and demonstrate how it overcomes specific challenges.

---

\(^2\) You can learn much more about the entire PTS family, with details on the physical units (i.e., PTS 22000, PTS 24000, and the PTS 32000) and the PTS Virtual Series, at [https://www.sandvine.com/platform/policy-traffic-switch.html](https://www.sandvine.com/platform/policy-traffic-switch.html)

\(^3\) Technically, core affinity can extend beyond simple flows, sessions, and subscribers to any ‘balanced set’. The balanced set is at its minimum a single subscriber IP address. At its most complex the balanced set can be a grouping of subscribers, with multiple IP addresses each, who all belong to some sort of collection. The collection can be a physical mapping (e.g., a CMTS QAM grouping or a mobile cell) or it can be a virtual mapping (e.g., all gold tier subscribers in a zip/postal code). For the sake of brevity, though, this document will cut off the list at “flow, session, and subscriber”. 
Ensuring Core Affinity to Overcome Routing Asymmetry

With clustering, the packet flow remains unchanged at the network level: it enters the cluster via whatever link it appears on the network, and it exits onto the same link. Internally, though, regardless of the link on which a particular packet arrives at the entire cluster, that packet is carried to a specific processor core.\(^\text{4}\)

From the perspective of a particular processor core, packets are always presented symmetrically, in the correct order. The result is that asymmetry is overcome. This behavior is maintained regardless of hardware cabling and routing asymmetry in the network.

The important consequence of this achievement is that all functions and features of the PTS are available, regardless of the degree of asymmetry on the network. Perhaps more than anything, this characteristic is what makes clustering the superior (and only viable) way to apply stateful network policy control to asymmetric traffic. Alternatives (e.g., state-sharing) require a CSP to sacrifice functionality and vary in effectiveness based on network characteristics; with clustering, everything works in every single scenario.

But how do the packets get directed to the right processor?

The Network Processing Unit

To ensure core affinity, Sandvine has created a network processing unit (NPU). The NPU is the first point of examination for incoming packets on a Policy Traffic Switch, and is dedicated to maintaining flow, session, and subscriber affinity for maximum element throughput. Figure 2 shows a simplified view inside a PTS, and shows the ingress and egress NPUs\(^\text{5}\).

---

\(^{4}\) The importance of core affinity is explained in great detail in the Sandvine whitepaper *QuickPath Interconnect: Considerations in Packet Processing Applications*

\(^{5}\) The ingress and egress are on the same interface
Incoming packets are first examined by the ingress NPU to determine whether the traffic even needs to be inspected (i.e., passed to a CPU). For example, depending on the policy, traffic belonging to certain VLANs may not be inspected, which may be desired if the service provider chooses not to inspect traffic that belongs to a wholesale customer or business customer. If the traffic does not need further inspection then it exits straight through the egress NPU. This scenario is illustrated by the blue/dashed line in Figure 3.

For those packets that should be sent to a CPU (i.e., a PPU in the case of the PTS), the NPU creates and relies upon a map that determines which processor core will process particular flows, sessions, and subscribers, and directs the packets appropriately. This mapping ensures that the same core is always used for all packet-processing relating to a specific flow, session, and subscriber. The map scales by the number of cores in the system, not the number of packets per second, so performance is not impacted. The path of a packet that goes to a PPU is illustrated by the red/dotted line in Figure 3.

Figure 3 - Packet flow on a single PTS: blue/dashed is a packet with no further processing; red/dotted is a packet that goes to a Policy Processing Unit (PPU)
If that core is on the box at which the packet arrived, then the redirection is carried out on switching fabric internally; if that core is on a different box, then the packet traverses the cluster links\(^6\) (i.e., the 10GE Stacking Ports in Figure 2) to arrive at the box housing the core. Once processed, the packet is returned to the exit interface corresponding to the original link, making clustering transparent to the network.

![Figure 4 - Packet flow when a packet enters one PTS but is destined for a PPU in another](image)

Clustering completely, rather than approximately, resolves all functional issues that complicate applying stateful network policy control to asymmetrically routed traffic and ensures complete consistency with a fully symmetric deployment - everything works: policy enforcement, measurements, VAS-enablement, etc.

Critically, clustering works in any deployment, with any number of asymmetric links\(^7\), with any forms of traffic\(^8\) - that means CSPs are able to make network changes without concern that the policy control will be impacted.

### Latency Considerations

The most common concern raised about the clustering approach is about the latency that is added by moving packets between PTS units.

In a PTS cluster, all the PTS units are connected via a full mesh via cluster cables. Because of the full mesh, every PTS is exactly one hop away every other PTS. If we assume conditions such that a packet is equally likely to be destined for any one of the PTS units, then we have the following:

- There is a \(1/N\) chance that a packet is processed on the PTS unit that it physical entered: in this case, the packet’s path is the red/dotted line in Figure 3: Ingress NPU→Switch Fabric→PPU→Switch Fabric→Egress NPU
- There is an \((N-1)/N\) chance that a packet is processed on a different PTS, and that PTS is guaranteed via the full mesh to be one hop away: in this case, the packet’s path is the

\(^6\) From a deployment complexity standpoint, clustering is no more or less complex than state-sharing, as both approaches require connecting multiple units together. The cost incurred in providing the larger cluster links is more than offset by the increased efficiency, guarantee of functionality, and preservation of the CSP’s ability to make routing changes to the network

\(^7\) For contrast, state-sharing typically does not work when there are more than two links being intersected

\(^8\) State-sharing approaches also typically don’t work with UDP traffic
red/dotted line in Figure 4: Ingress NPU → Switch Fabric → Cluster Link → Switch Fabric → PPU → Switch Fabric → Cluster Link → Switch Fabric → Egress NPU

We can see that when a packet needs to hop to a different PTS, this adds two passes through the switch fabric and two through the cluster link.

Recall that all of these calculations assume that a packet must be examined by the PPU, and this is not the case; many packets do not require inspection or processing, and these packets simply go Ingress NPU → Egress NPU (i.e., the blue/dashed line in Figure 3).

The reality is that per-packet latency varies based on paths and policy application⁹, but we can nevertheless provide some numbers. For instance, with a PTS 22600 running PTS 6.40 software, tests show that the minimum latency through a single unit is around 40μs and the median is around 180μs. The maximum is 340μs and 99.9% of packets experience a latency of less than 300μs.

Combining these figures with reasonable assumptions for latency of the cluster cable and switch fabric (we will assume 10μs for the extra passes¹⁰) yields the latencies presented in Table 1.

<table>
<thead>
<tr>
<th>Cluster Size</th>
<th>Expected Latency (μs) Per Packet Assuming Uniform PTS Probability</th>
<th>&lt;99.9%</th>
</tr>
</thead>
<tbody>
<tr>
<td>2*PTS</td>
<td>Minimum: 49.00, Median: 185.00, Maximum: 345.00, &lt;99.9%: 305.00</td>
<td></td>
</tr>
<tr>
<td>3*PTS</td>
<td>Minimum: 50.67, Median: 186.67, Maximum: 346.67, &lt;99.9%: 306.67</td>
<td></td>
</tr>
<tr>
<td>4*PTS</td>
<td>Minimum: 51.50, Median: 187.50, Maximum: 347.50, &lt;99.9%: 307.50</td>
<td></td>
</tr>
<tr>
<td>5*PTS</td>
<td>Minimum: 52.00, Median: 188.00, Maximum: 348.00, &lt;99.9%: 308.00</td>
<td></td>
</tr>
<tr>
<td>6*PTS</td>
<td>Minimum: 52.33, Median: 188.33, Maximum: 348.33, &lt;99.9%: 308.33</td>
<td></td>
</tr>
<tr>
<td>7*PTS</td>
<td>Minimum: 52.57, Median: 188.57, Maximum: 348.57, &lt;99.9%: 308.57</td>
<td></td>
</tr>
<tr>
<td>8*PTS</td>
<td>Minimum: 52.75, Median: 188.75, Maximum: 348.75, &lt;99.9%: 308.75</td>
<td></td>
</tr>
<tr>
<td>9*PTS</td>
<td>Minimum: 52.89, Median: 188.89, Maximum: 348.89, &lt;99.9%: 308.89</td>
<td></td>
</tr>
</tbody>
</table>

In this overwhelmingly more common scenario, the average latency from the moment a packet enters the cluster to the moment it leaves (i.e., accounting for all inspection and any inter-unit switching within the cluster) is less than 200μs.

The only reason the latency increases ever-so-slightly as we increase the number of PTS units in the cluster is that as we add more PTS units there is a slightly decreased likelihood that the packet will be processed on the physical PTS that it entered. If we were to assume 1μs for the additional switch fabric and cluster cable passes instead of 10μs, then the rows would show a latency variance of only a few nanoseconds.

The point can be summarized quite succinctly: clustering adds no meaningful packet latency.

---

⁹ And also many other factors including, but not limited to: traffic mix, PTS model (i.e., different processors), and PTS software (newer releases generally have higher performance optimization). It’s important to note that the same factors will create variance in any network policy control device.

¹⁰ This is a very conservative/high assumption - consider that the latency of the packet passing twice through a 1m cluster cable is 0.01μs, which is 10 nanoseconds.
Clustering to Achieve Efficient Scale and Redundancy

Any network policy control solution must be able to achieve massive scale\textsuperscript{11} to be viable in today’s communications networks. A PTS cluster doesn’t just overcome routing asymmetry, it also delivers tremendous scale.

Ultimately, the maximum cluster scale is a function of the number of cluster links available on the individual PTS units. At the time of writing, the largest possible PTS cluster achieves inspection throughput of 8 Tbps.

But absolute scale isn’t the only important scaling characteristic of PTS clusters; with issues of scale come questions about efficiency (i.e., performance density), extensibility (i.e., is it linear, or does it suffer from diminishing returns), and redundancy (i.e., what happens when something goes wrong).

Performance Density and Extensibility

The NPU is of vital importance to packet processing and policy control performance in both micro and macro terms: on the micro level, the NPU maximizes performance of every single core and PPU; at the macro level, the NPU-based architecture means that every additional core, PPU, and even PTS device adds linearly to the overall processing capacity of the entire PTS cluster.

This latter characteristic should not be understated - it is extremely rare in computing that the addition of processing capacity is achieved without diminishing returns.\textsuperscript{12} By contrast, since units in a state-sharing architecture share state across boxes, each additional unit added to a deployment actually decreases the state memory available to each box.

Practically, the shared nothing architecture means that a PTS cluster exhibits two important characteristics:

- **Performance density is maintained when units are added:** The per-rack-unit performance metrics don’t change when additional PTS are introduced to the cluster. In the 8 Tbps PTS cluster mentioned previously, performance density is 100 Gbps per rack unit.
- **Extensibility is linear:** Each PTS that is added to a cluster adds to that cluster 100% of the new unit’s processing capacity. In this manner, a PTS cluster is able to easily and efficiently accommodate network growth.

The shared nothing architecture also has some important consequences for the economics of redundancy.

Redundancy

Because there is no shared state between PTS units, the redundancy model is N:N+1. That is, if the deployment requires a cluster of four PTS units to deliver the required performance, then the redundant deployment requires five.

In an architecture that shares anything, there must be a copy of each piece. Impractically, this leads to a redundancy model of N:2N. That is, if the performance requirements only warrant four units, the redundancy model demands eight.

\textsuperscript{11} “Scale” can take on many meanings, including inspection throughput (measured in bandwidth), concurrent subscribers, concurrent flows, new flows per second, etc. In this paper, we’ll just stick with bandwidth, but the examination and conclusions are equally applicable to all the scaling factors.

\textsuperscript{12} You can learn more about this subject by reading [http://en.wikipedia.org/wiki/Amdahl’s_law](http://en.wikipedia.org/wiki/Amdahl’s_law)
The N:N+1 model is obviously preferable from an economics standpoint, whether measured in terms of capital outlay or in terms of ongoing expenses associated with maintaining hot rack-space for the standby units.

In the Sandvine cluster, rather than keeping the “+1” unit sitting idle, it becomes part of the cluster and participates in the processing. In the event of core, processor, or unit failure, the traffic is simply rebalanced to the remaining N units (recall that the performance requirements called for N units, so N is enough to accommodate the fail-over). When the failed element is incorporated back into the cluster, it re-assumes its share of traffic.
Conclusion

To be viable solution platforms in modern communications networks, stateful packet-processing equipment must overcome the challenges posed by widespread routing asymmetry and immense scale.

Policy Traffic Switch clustering overcomes the practical limitations of the “one giant box” approach by emulating the behavior of that single box across many in order to overcome asymmetry, deliver extreme scale, and preserve redundancy; importantly, clustering achieves these goals with incredible efficiency. The shared nothing architecture ensures linear extensibility and enables an economical N:N+1 redundancy model.

Importantly, a PTS cluster is managed as a single unit, so there is no added operational management complexity, and does not introduce any meaningful latency (refer to Table 1). To further simplify life for the operations team, the cluster is self-configuring and self-managing.

Critically, PTS clustering overcomes asymmetry and delivers scale in any access network, with any combination of access technologies, and completely independent of the network’s routing - this means that the communications service provider is free to make changes to the network without fear of impacting the network policy control deployment.

Alternatives (e.g., state-sharing) require a CSP to sacrifice functionality and vary in effectiveness based on network characteristics; with clustering, everything works in every single scenario.

PTS clustering is a beautiful, elegant, and - above all - effective and efficient solution to the challenges imposed by routing asymmetry and the demands of massive scale.

Related Resources

More information about the subject of routing asymmetry and its implications for stateful network policy control is available in the Sandvine whitepaper Applying Network Policy Control to Asymmetric Traffic: Considerations and Solutions.