

WHITE PAPER

From Signal to Certainty:

Building Network Observability on Packet Truth

Executive Summary

Modern enterprise networks are increasingly distributed across on-premises infrastructure, public and private clouds, SaaS platforms, and remote sites. Monitoring the health, security and service availability across such [complex environments](#) requires telemetry from a network, application, and external providers. But monitoring alone is not enough. Many network operations (NetOps) teams are shifting to more integrated observability stacks – providing greater visibility, intelligence, and scale to improve business outcomes. This requires a holistic data foundation comprising multiple data sources.

MELT (which includes metrics, events, logs, and traces) and flow data enable network operation teams to observe workloads at scale. By design, they summarize, sample, or instrument behavior, rather than record everything. While important, MELT and flow cannot definitively explain what happened on the network, which may be insufficient for analyzing root cause, diagnosing intermittent performance issues, or reducing Mean Time to Resolution (MTTR) across multiple teams.

Packet data catches granular details that reveal what other tools miss. In addition to showing exactly what actually happened on the wire, packet analytics objectively measure the three dimensions of user experience that matter most for root cause analysis: latency, connectivity, and quality. Just as importantly, packet data provides indisputable evidence that stands up to operational and compliance scrutiny.

As IT organizations evaluate their network observability stacks, many are learning a valuable lesson. MELT and flow telemetry add speed, scale, and context – but yield the best outcomes when anchored in packet truth.

This paper examines the strengths and limitations of each observability data type and explains how enterprises can adopt packet-centric observability architectures to complement and bolster their existing tools. It also provides four real-world scenarios that highlight when data can be used in tandem to achieve better, faster outcomes.

Section 1: Types of Observability Data

1.1 Packets: The Foundational Observability Layer

Observability without packet data is observability built on assumptions and guesswork. Packet capture records every packet that traverses the network, including headers, payloads, and precise timing. Unlike other observability signals, packets are not derived, summarized, or dependent on instrumentation. They represent what was actually transmitted and received, in the order it occurred.

Data that stands up to scrutiny

Similar to evidentiary standards in a courtroom, a recorded conversation is always more believable than a witness recounting the same conversation. Confidence in the evidence is increasingly important as organizations adopt AI to automate and improve operations. Without trust in the data foundation, finger-pointing ensues, escalations grow, and projects stall.

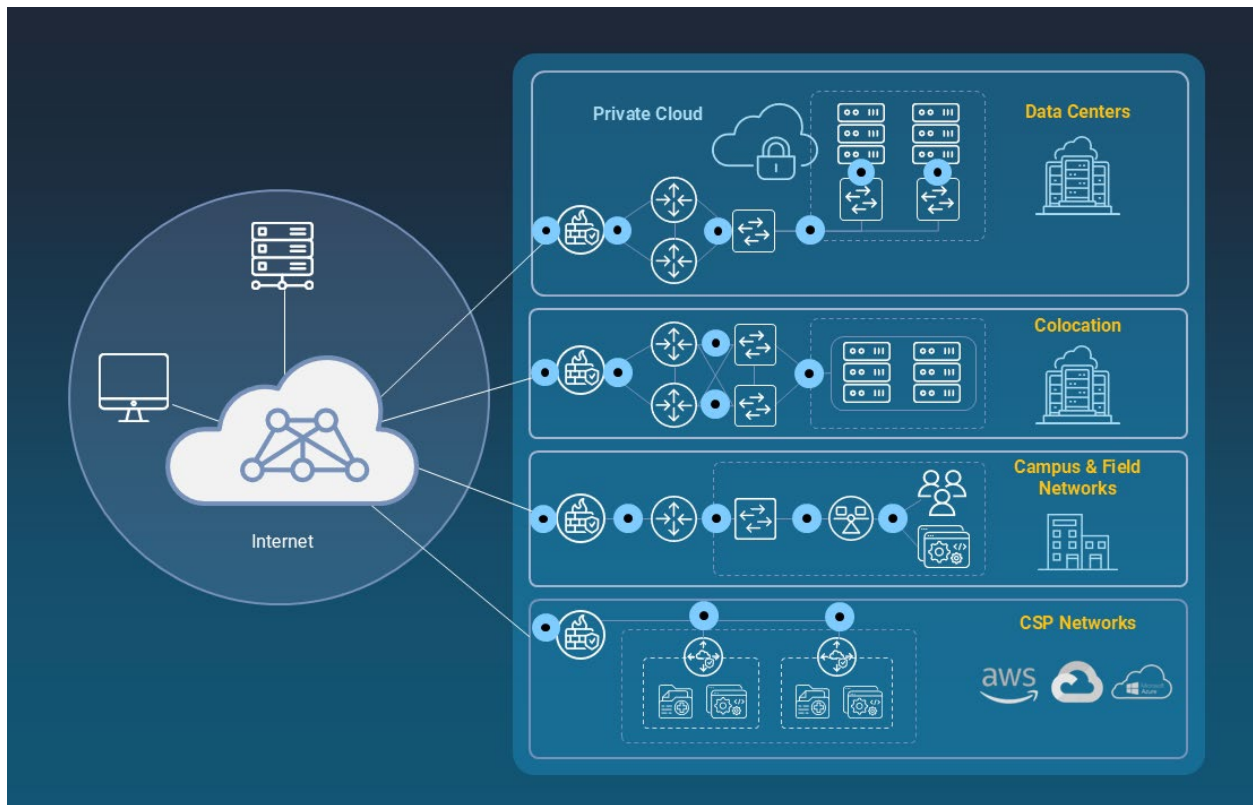
Packet analytics measure individual packets, maintain full connection state, and preserve protocol semantics. This enables precise visibility into how applications behave over the network and where failures occur — whether in the network itself, a load balancer, a firewall, or an endpoint.

Because packets are captured independently of applications and infrastructure components, they provide an objective, tamper-resistant record of events. They do not rely on observability agents (such as OpenTelemetry), developer instrumentation, or flow sampling, and they remain reliable even when systems are misconfigured or compromised.

The granularity and end-to-end pipeline of packet analysis can be used to measure all three dimensions of user experience:

- **Latency** (a.k.a. slow service), including round-trip time, server response time, and delay at each hop
- **Connectivity**, including connection establishment, authentication flows, resets, and failures
- **Quality**, including packet loss, jitter, retransmissions, and out-of-order delivery

Packet acquisition does have practical constraints. Capturing and storing trillions of raw packets each day can be resource-intensive and expensive. Effective packet observability therefore requires strategic placement, selective capturing with filters, and intelligent indexing once those packets are captured. These are architectural considerations rather than fundamental limitations.



Practical Guidance:

Packets are essential when organizations need to measure actual user experience, determine where latency is introduced, debug protocols and applications, investigate security incidents, or establish objective evidence during disputes or audits. They are always required when issues are intermittent, performance-related, or contested across teams.

1.2 Metrics: Resource Utilization, Not Experience

Metrics describe the state of systems and infrastructure at a single point in time. Host-level metrics report CPU, memory, disk, and interface utilization. Network device metrics, typically collected via SNMP, report counters such as bytes transmitted, errors, and drops.

While metrics are excellent for understanding resource consumption and long-term trends, they do not measure how users experience applications. They are typically reported at coarse intervals and are stateless, meaning they cannot correlate resource usage with specific transactions or failures.

In high-speed networks, short-lived microbursts that cause packet loss or latency spikes can be completely invisible in minute-level averages – thus potentially mask the true issue. Metrics may indicate that a system was busy, but not that specific packets were dropped or ignored.

Practical Guidance:

Metrics are most effective for capacity planning, detecting sustained overload, and monitoring overall system health. They are not suitable for diagnosing intermittent failures, sub-second performance issues (of which there are many), or transaction-level root causes.

1.3 Logs and Events: Context and Intent, but Subjective and Incomplete

Logs provide insight into what applications and systems *believe* happened. They record user actions, configuration changes, and events – when something changes state or crosses a predefined threshold. Together, they alert users and provide important context, error messages, and transaction identifiers. System and security logs also feed Security Information and Event Management (SIEM) platforms and compliance workflows.

However, logs and events only reflect what developers or system designers anticipated and chose to capture. From a user-experience perspective, they can measure the latency of communication for an internal machine state – but only if developers intentionally add instrumentation. Even then, coverage is limited to known code paths. Logs cannot independently verify network behavior and can be altered or deleted if a system is compromised.

Practical Guidance:

Events are most effective as early-warning indicators and correlation signals, while logs answer critical questions about *why* a system behaved a certain way and *who* initiated a change. They are valuable for understanding intent and internal state, but are inherently subjective and incomplete. This is insufficient for root cause analysis when latency, connectivity, or packet-level quality issues are involved – especially when failures were not anticipated by developers.

1.4 Traces: Application Latency with Limited Scope

Distributed tracing instruments application code to track requests as they traverse microservices. Traces are highly effective for understanding latency inside applications and identifying slow code paths.

However, traces only measure what is instrumented. They do not capture network latency between services, nor do they provide visibility into un-instrumented infrastructure such as load balancers, firewalls, DNS, or third-party APIs. Traces also cannot measure packet loss, retransmissions, or connectivity failures.

Practical Guidance:

Because instrumentation requires ongoing development effort and introduces runtime overhead, trace coverage is often incomplete. Traces are best suited for application-level performance analysis, not end-to-end user experience or network troubleshooting.

1.5 Network Flows: Traffic Volume Without State

Flow telemetry summarizes network conversations between endpoints typically using 5-tuple aggregation – source IP, destination IP, source port, destination port, and IP protocol. Flows report how many bytes and packets were exchanged, over what period of time, and through which interfaces.

Flow data scales efficiently and is useful for understanding traffic patterns, identifying large data transfers, and establishing baselines. However, flows are stateless. They do not track connection success, latency, or quality. They cannot explain why transactions failed or why applications were slow.

Practical Guidance:

Flows are valuable for navigation and trend analysis, but insufficient for analyzing network health and the root cause of incidents. Organizations often use flow data to focus on “top talkers” because that is what flows measure best. This can mislead investigations toward volume rather than root cause.

1.6 Synthetic Monitoring: Simulated Visibility

Synthetic monitoring tools inject test traffic to simulate user behavior and measure performance from specific vantage points. They are effective for proactive testing of known paths, such as branch-to-datacenter or user-to-SaaS connectivity.

However, synthetic tests are simulated, not real. They only measure predefined workflows and do not capture unexpected or intermittent issues. Synthetic tools are also primarily designed for north-south traffic and offer limited visibility into east-west, server-to-server communication.

Practical Guidance:

Synthetic monitoring complements real traffic analysis but cannot replace it for incident investigation.

Section 2: Packet-Anchored Observability Workflows

Each data source answers different questions.

Effective observability workflows use abstracted data (metrics, flows, synthetic tests) to detect and triage at scale, then pivot to packets for definitive root cause analysis and evidence-based resolution.

Data Source	Role in Workflow	What it Provides
Synthetic Monitoring	Early Detection	Path availability, baseline latency from known endpoints
Metrics	Resource Visibility	CPU, memory, disk, interface utilization trends
Logs & Events	Change Context	Timestamped state changes, configuration changes, and user actions
Flow Data	Traffic Patterns	Volume anomalies, communication patterns, bandwidth usage
Packets	Ground Truth	Precise timing, connection state, protocol behavior, location of fault

The following user stories demonstrate how to leverage multiple types of telemetry to accelerate detection, troubleshooting, and resolution in common network and security scenarios.

Scenario 1: A regional sales office reports intermittent application connection failures affecting Virtual Desktop Infrastructure (VDI) access.

Detection (Synthetic Monitoring): The network operations team receives an alert from their synthetic monitoring platform showing elevated latency on the path between the branch office and the data center. Test packets confirm connectivity exists, but round-trip times increase from 45ms to 300ms+ between 8AM and 9AM every workday. This increased latency causes intermittent authentication failures.

Investigation:

- Network engineers use packet capture and analytics at the data center and in a colocation edge data center entry point.
- Comparing capture points, the team identifies the WAN optimization appliance as the source of added latency.
- Packets show data entering the WAAS at normal timing, but exiting with significant delay, while return traffic flows normally.

Resolution: The WAAS logs show the device is entering a degraded state due to disk I/O contention from an oversized deduplication dictionary – resulting from large backup operations that were scheduled to 8AM instead of midnight. Restarting the optimization service and adjusting dictionary size parameters and redirecting the backup job restores normal performance.

Key Insight: Synthetic monitoring detected the symptom across the entire path. Packets identified the exact device and direction causing the problem – information that is impossible to derive from end-to-end latency measurements alone. Logs and flows helped with understanding the exact root cause.

Scenario 2: Customers report a business-critical ITSM application has been sluggish

Detection: A customer using a mission-critical SaaS application reports slow access to the application, which started the previous day.

Investigation:

- The metrics from the packet capture and analytics showed added latency but revealed that it was caused by application servers that became sluggish around midnight.
- Change management logs confirm a scheduled patch was applied to the application at 11:45 PM. A search of the vendor's knowledge base and user forums reveals multiple customers reporting similar performance degradation after installing this specific patch version, with the vendor acknowledging a memory leak in the connection pooling module.
- Server metrics show increased memory and CPU consumption following the patch application.

Validation: Before rolling back, the team uses packet capture at the server interface to determine if the patch is affecting user experience – not just consuming resources. Analysis reveals:

- Server response times increased from 15ms average to 180ms average
- TCP window scaling shows the server advertising smaller receive windows under memory pressure
- Retransmission rates increased 4x as the server struggles to acknowledge data promptly

Resolution: Armed with packet evidence showing direct user impact, the team receives approval to roll back the patch during a brief maintenance window rather than waiting for the vendor's fix. Post-rollback packets confirm response times returned to baseline.

Key Insight: Metrics showed resource consumption. Logs identified what changed and when. External research confirmed others experienced the same issue. Packets proved the resource problem translated to degraded user experience – justifying the urgent rollback decision.

Scenario 3: A financial services firm experiences intermittent connection failures to a critical trading API after scheduled maintenance.

Detection: Users complained about failing to connect to a trading application, errors affect approximately 10% of connection attempts, with no clear pattern in which clients or transactions fail.

Investigation: Packet capture on the user's side showed a distinct pattern in failing connections:

- SYN → SYN-ACK → ACK → Client Hello → RST
(Explanation: The TCP handshake succeeds, but the connection is immediately reset after the TLS Client Hello. The server logs showed no indication of a failed connection, nor did any of the other devices along the path show any errors.)
- Comparing the packet capture in different locations revealed that the connection reject is initiated by the load balancer. Using the packet capture to determine when it started, it was aligned with a load-balancer software upgrade.

Root Cause (Vendor Knowledge Base): A search of the load balancer vendor's release note confirms that the new firmware version enforces SNI validation by default – a security hardening feature that the vendor considers beneficial. Connections without SNI are rejected as potentially malicious.

Resolution: The team has two options: configure the load balancer to allow connections without SNI, or update legacy clients to include SNI. They choose to temporarily disable SNI enforcement for the trading API virtual server while coordinating client updates.

Key Insight: Logs couldn't explain why 15% of connections—all seemingly identical—were rejected. Packets revealed the specific TLS handshake difference, and vendor documentation confirmed the behavioral change. Without packets, the team would have been debugging blind.

Scenario 4: Using analysis of deep packet inspection, the security team observes unusual external DNS query patterns during routine traffic analysis.

Detection: Continuous packet monitoring flags an anomaly: external DNS queries started to use thousands of external DNS public resolvers over the past 48 hours. Previously, internal clients sent queries exclusively to corporate DNS servers, which handled recursion.

Security Concern: This "DNS sprawl" creates multiple risks. It not only bypasses corporate DNS security policies (malware domain blocking, content filtering), but exposes internal hostnames in external queries. This could indicate malware using DNS for command-and-control communication.

Investigation:

- The security team correlates the timing with Active Directory change logs.
- They discover that during a routine AD maintenance window, an administrator inadvertently removed the upstream recursive resolver configuration from the corporate DNS servers.
- With no upstream forwarders configured, DNS server queries external domains.

Validation: Packet analysis confirms the hypothesis.

Resolution: The team restores the upstream resolver configuration. Within minutes, packet monitoring confirms external DNS queries drop to near-zero as corporate servers resume handling recursive lookups.

Key Insight: Packets detected the behavioral anomaly – something that wouldn't trigger alerts in logs or metrics because each individual query was technically successful. Logs explained *why* the change occurred. Together, they resolved what could have been treated as a security incident in under an hour.

Section 3: Decision Framework for Enterprise Observability

Operational decisions are driven by the specific outcomes teams need to achieve. Different use cases – such as troubleshooting application performance, ensuring service availability, investigating security anomalies, or validating compliance – require different types of telemetry and analytical approaches. The value of any telemetry source therefore depends on the operational question being asked and the metrics that define success for that scenario.

In most cases, critical operational questions – how slow an application is, where latency occurs, whether connections succeed, and why quality degrades – can only be answered definitively with packet data. But other observability sources play important roles:

- Metrics describe resource state
- Events indicate something happened in a window of time
- Logs provide intent and context
- Traces illuminate application internals
- Flows scale visibility across environments
- Synthetic monitoring offers proactive testing

Requirement	Packets	Flow	MELT
Application Insight	Moderate	Weak	Strong
Accurate network performance	Strong	Limited	Weak
Root cause certainty	Strong	Weak	Weak
Security anomaly detection	Strong	Strong	Moderate
Forensic evidence	Strong	Moderate	Weak
Dependency on instrumentation	None	Medium	High

Therefore, the recommended architecture is a tiered observability model that uses multiple data sources:

- Packet data as the foundation for truth and proof – when captured strategically
- Flow telemetry for scale and navigation
- MELT data for speed and application context

Conclusion

As enterprise networks become increasingly complex, NetOps teams should not be forced to decide between abstracted telemetry and packet truth. The future of network observability requires a balance between speed, quality, and scale.

MELT and flow data helps engineers detect incidents and narrows their investigation with context. Packet data provides the objective record required to resolve incidents, eliminate ambiguity, and satisfy regulatory and security demands.

The future of network observability, but in building abstractions on top of packet truth. This is what enables faster resolution, reduced operational friction, and greater confidence in both performance and security decisions.

cPacket Networks provides deep packet observability solutions that give IT operations teams complete visibility into network performance, security, and application behavior. Learn more at <https://www.cpacket.com>.