WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

NETWORK TRAFFIC ANALYSIS AND PERFORMANCE EVALUATION USING WIRESHARK: A REVIEW

Dhirendra Kumar[1], Manish Kumar Soni[2]

[1] Author / Corresponding Author:
[2] Guide Name
BANSAL INSTITUTE OF ENGINEERING & TECHNOLOGY, APJ ABDUL KALAM TECHNICAL UNIVERSITY AKTU
BANSAL INSTITUTE OF ENGINEERING & TECHNOLOGY, APJ ABDUL KALAM TECHNICAL UNIVERSITY AKTU

ABSTRACT :

Network traffic analysis is a fundamental activity in network administration, troubleshooting, security monitoring, and performance evaluation. Modern applications—including cloud services, online learning platforms, multimedia communication, and Internet of Things (IoT) systems—depend on predictable communication performance. Wireshark is an open-source protocol analyzer that supports live packet capture, protocol dissection, display filtering, and statistical inspection of captured traffic. This review examines how Wireshark can support network traffic characterization and performance analysis, with particular attention to throughput, round-trip time (RTT), packet loss, delay variation, bandwidth utilization, and TCP retransmissions. The paper synthesizes the capabilities described in Wireshark's official documentation with established Internet Engineering Task Force (IETF) guidance on network-device benchmarking and TCP throughput testing. It proposes a reproducible workflow that separates packet observation from active performance measurement and explains why packet captures alone cannot establish every end-to-end performance metric. The review identifies practical limitations associated with capture location, packet drops, encryption, timestamp accuracy, and interpretation of analyzer-generated TCP flags. It concludes that Wireshark is most effective when combined with controlled traffic generation, receiver-side measurements, repeated trials, and explicit metric definitions. Opportunities for future work include automated traffic classification, anomaly detection, encrypted-traffic characterization, and integration with complementary monitoring tools.

Keywords: *Wireshark, network traffic analysis, network performance, packet capture, throughput, round-trip time, packet loss, TCP retransmission, network monitoring.*

1. Introduction

Computer networks provide the communication infrastructure for web applications, cloud platforms, enterprise systems, online education, video conferencing, and IoT services. The quality of these applications is influenced by network capacity, delay, congestion, packet delivery, protocol behavior, and endpoint processing. When users experience slow transfers or intermittent service, aggregate interface counters may show that a problem exists without explaining the communication events associated with it.

Network traffic analysis examines packets and communication exchanges to characterize protocol use, endpoint behavior, timing, packet sizes, and transport-layer events. Packet-level evidence can complement higher-level metrics by revealing how a particular connection behaves during a transfer. Wireshark provides protocol decoding, capture-file inspection, display filters, conversation statistics, I/O graphs, and protocol-specific analysis features (Wireshark Foundation, n.d.). These capabilities make it useful in networking laboratories, troubleshooting, and controlled performance investigations.

However, packet inspection and performance benchmarking are related but distinct tasks. A capture describes traffic visible at a particular observation point. It does not automatically measure the maximum capacity of a link, establish one-way latency, or prove that a missing packet was lost in the network. IETF benchmarking guidance emphasizes defined test conditions, repeatability, and explicit reporting (Bradner & McQuaid, 1999), while its TCP throughput framework describes the importance of baseline RTT, bottleneck bandwidth, path MTU, and TCP window behavior (Constantine et al., 2011).

1.1 Problem statement

The central problem addressed by this review is how to use Wireshark to derive meaningful evidence about network behavior while avoiding invalid conclusions caused by incomplete captures, ambiguous metric definitions, encryption, or uncontrolled test conditions.

1.2 Objectives

- Review Wireshark features relevant to packet-level network traffic analysis.
- Explain the measurement meaning and limitations of common performance metrics.
- Propose a reproducible workflow for Wireshark-assisted evaluation.
- Compare packet-analysis capabilities with controlled network benchmarking.
- Identify research opportunities in automation, anomaly detection, and encrypted traffic analysis.

1.3 Review scope

The paper focuses on IP networking and observable TCP, UDP, DNS, HTTP, TLS, and related traffic. It is a narrative technical review based on official documentation and standards. It is not claimed to be a systematic literature review: a formal multi-database search, screening protocol, and quality assessment have not been conducted. No original packet-capture experiment is claimed, and no fabricated empirical results are presented.

2. Review approach and source selection

The review is organized around four themes: (i) packet capture and protocol inspection, (ii) performance metrics and their interpretation, (iii) experimental design and measurement reliability, and (iv) limitations and future directions. The primary technical sources are the Wireshark User's Guide (Wireshark Foundation, n.d.), RFC 2544 (Bradner & McQuaid, 1999), and RFC 6349 (Constantine et al., 2011). Peer-reviewed sources were selected for their direct relevance to packet-level diagnosis, monitoring-tool comparison, wireless performance measurement, and encrypted-traffic analysis. This is a targeted narrative review, not a systematic review; no claim is made that every relevant publication has been identified.

2.1 Literature review and synthesis

Research on packet-level network analysis spans practical monitoring tools, application performance diagnosis, network benchmarking, and traffic-trace characterization. The literature is not uniform in purpose: some works use packet captures to diagnose a particular application, others compare monitoring products, and standards documents define controlled test procedures. Recent surveys also examine traffic analysis when payloads are encrypted, including machine-learning approaches that use flow and timing features (Shen et al., 2023; Velan et al., 2015; Alwhbi et al., 2024). These contributions should therefore be compared by their research question and measurement method rather than treated as interchangeable evidence.

Karamollahi et al. (2023) studied packet-level traces of Zoom performance anomalies and used traces collected from different vantage points to identify a congested external Internet link as the cause of reported campus performance problems. Their work demonstrates the value of multiple observation points and packet-level evidence for diagnosing an application-level symptom; it also illustrates why a single capture location may not be sufficient to localize a fault.

Alizan and Zolkipli (2024) compared Wireshark with Paessler Router Traffic Grapher (PRTG), distinguishing Wireshark's packet-level troubleshooting strengths from PRTG's broader device-monitoring and scalability focus. This comparison is relevant to tool selection, but it does not mean that the two tools measure identical quantities: packet inspection and long-term infrastructure monitoring serve complementary roles.

Soewito et al. (2017) examined simulated file transfers before and after WAN optimization using Wireshark-based measurements. The publisher abstract reports a change in delay from 287 ms to 0.604 ms for a 93 MB file and an improvement in jitter. These are findings from that study's particular setup and workload; they should not be interpreted as a general performance gain guaranteed by Wireshark or by WAN optimization in every network.

Birahmatika et al. (2024) compared Wireshark and PRTG for wireless-network quality-of-service analysis over seven days. The paper reports Wireshark measurements of 1.418 Mb/s throughput, 2.146% packet loss, 4.095 ms delay, and 4.08 ms jitter, while PRTG reports 2.231 Mb/s throughput, 0.493% packet loss, 3.094 ms delay, and 4.12 ms jitter. These values describe the authors' observation environment and measurement procedure; they should not be generalized to other networks or treated as a universal ranking of the tools.

Sikos (2020) surveyed packet analysis for network forensics, including packet analyzers, deep packet inspection, traffic classification, and AI-assisted methods. Although the paper's primary focus is forensic analysis rather than performance benchmarking, it provides broader context for packet-analysis capabilities and highlights the importance of capture completeness, tool characteristics, and privacy.

Shen et al. (2023) reviewed machine-learning-powered encrypted network traffic analysis and organized prior work by goals such as network asset identification, network characterization, privacy-leakage detection, and anomaly detection. Their survey shows that encryption changes the evidence available to analysts: payload inspection becomes less effective, while traffic representation and feature selection become central parts of the analysis workflow. This literature is relevant to Wireshark-based work because packet capture can supply metadata and timing observations, but classification models require separate training, validation, and performance assessment.

Alwhbi et al. (2024) surveyed machine-learning approaches for encrypted network traffic analysis and classification, discussing the workflow and methods used to infer traffic characteristics without directly inspecting plaintext payloads. Together with Velan et al. (2015), these surveys indicate that encrypted-traffic classification is a distinct analytical problem from measuring throughput or packet loss. A review of Wireshark-based performance evaluation should therefore avoid implying that packet capture alone reveals encrypted application content or establishes application identity.

Taken together, these sources suggest four complementary strands: (i) packet-level diagnosis of application problems, (ii) comparison of monitoring tools and measurement practices, (iii) controlled benchmarking, and (iv) broader surveys of packet-analysis techniques. A recurring methodological issue is that results depend on observation points, workload, test duration, and metric definitions. The present review synthesizes these strands into a measurement-oriented workflow, emphasizing the distinction between observed packet statistics and independently validated end-to-end performance.

Table 1. Selected literature on packet-level traffic and performance analysis

Study/source	Scope and evidence summarized	Critical implication
Karamollahi et al. (2023)	Zoom performance anomalies: Packet traces from multiple vantage points	Demonstrates packet-level diagnosis; findings are workload/topology-specific.
Alizan and Zolkipli (2024)	Wireshark vs PRTG: Comparative tool analysis	Clarifies complementary tool roles; not a controlled capacity benchmark.
Soewito et al. (2017)	Transfer performance after optimization: Wireshark-	Illustrates intervention evaluation; results should not be

Study/source	Scope and evidence summarized	Critical implication
	based measurements of file transfers	generalized without replication.
Birahmatika et al. (2024)	Wireless network performance and tool comparison: Throughput, loss, delay and jitter over seven days	Relevant metrics; comparability depends on configuration and calculation method.
Sikos (2020)	Packet analysis survey for network forensics: Survey of packet analysis, analyzers, DPI, and AI-assisted classification	Broad review context; not a direct performance benchmark.
Bradner and McQuaid (1999)	Network-device benchmarking: Controlled frame-based throughput and loss tests	Provides standardized test concepts; intended for laboratory benchmarking.
Constantine et al. (2011)	End-to-end TCP throughput: Baseline RTT/bottleneck bandwidth, TCP-window and transfer-time considerations	Supports repeatable TCP tests; packet capture alone is not the complete test.
Shen et al. (2023)	Encrypted traffic analysis: Survey of ML-powered traffic analysis	Useful for metadata-based classification; not a direct throughput benchmark.
Velan et al. (2015)	Encrypted traffic classification: Survey of classification methods and features	Explains observable features and limitations under encryption.
Alwhbi et al. (2024)	ML-based encrypted traffic analysis: Review of traffic classification methods	Supports future-work discussion; separate from direct performance measurement.

3. Fundamentals of Wireshark-based traffic analysis

3.1 Packet capture

Packet capture records packets visible to a selected network interface or monitoring point. Visibility depends on the network topology, interface mode, operating-system permissions, capture configuration, and whether the traffic traverses the selected interface. A capture taken on a client typically represents packets visible to that client, not all traffic in the local network. Capture loss may occur because of buffer limits, high packet rates, driver behavior, hardware offloading, or operating-system constraints. Capture statistics and configuration should therefore be retained with the dataset.

3.2 Protocol dissection

Wireshark decodes recognized protocol fields and presents packet details by protocol layer. This allows an analyst to inspect addressing, ports, TCP flags, sequence and acknowledgment information, DNS exchanges, and other protocol-specific fields. Dissection is descriptive rather than causal: for example, a retransmission indication can be consistent with loss, reordering, or other transport conditions. Diagnosis requires inspection of the surrounding sequence and, where feasible, evidence from another observation point.

3.3 Capture and display filters

Capture filters restrict which packets are recorded, while display filters select packets shown during analysis (Wireshark Foundation, n.d.). Display filters can focus an investigation on TCP, UDP, DNS, HTTP, TLS, an IP address, or a particular port. Illustrative filters include `tcp`, `udp`, `dns`, `tls`, `tcp.analysis.retransmission`, `tcp.analysis.duplicate_ack`, `ip.addr == 192.168.1.10`, and `tcp.port == 443`. Filters should be validated against the installed Wireshark version and the traffic actually present. Overly restrictive capture filters can remove evidence needed for later analysis.

3.4 Statistics and visual analysis

Protocol Hierarchy summarizes the protocols represented in the capture. Endpoint and Conversation statistics help identify traffic-intensive hosts and exchanges. I/O graphs visualize packet or byte counts over time, while packet-length statistics describe the size distribution of observed packets. These views are useful for identifying bursts, idle periods, dominant protocols, and high-volume conversations. They do not, by themselves, establish available link capacity or prove the cause of an application slowdown (Wireshark Foundation, n.d.).

3.5 TCP and application response analysis

Wireshark can identify TCP events such as retransmissions and duplicate acknowledgments, and it supports response-time statistics for selected protocols (Wireshark Foundation, n.d.). Such indicators can help direct investigation toward connection setup, transport recovery, or slow request-response exchanges. Response time is not necessarily pure network delay: it may include endpoint processing, queuing, forward-path delay, and return-path delay. Analyzer flags should be treated as evidence to investigate rather than as a definitive diagnosis.

4. Performance metrics and measurement considerations

4.1 Throughput and goodput

A basic byte-based traffic-rate estimate is $T = (8B)/\Delta t$, where B is the number of bytes counted under a stated convention and Δt is the measurement

duration in seconds. If 12,000,000 bytes are observed over 10 seconds, the resulting rate is 9.6 Mbps. This is an illustrative calculation, not a measured result. The meaning of B must be stated: captured bytes, IP-layer bytes, transport payload, and application-delivered data are not interchangeable. Goodput refers to useful application data delivered per unit time and excludes protocol overhead and retransmitted data that does not contribute new application payload.

Wireshark conversation statistics and I/O graphs can characterize observed traffic rates. Maximum achievable capacity requires a controlled test design and a suitable traffic generator/receiver. The TCP throughput testing framework recommends establishing baseline RTT and bottleneck bandwidth and considering path MTU and TCP window behavior when testing sustained TCP throughput (Constantine et al., 2011).

4.2 Latency and round-trip time

Latency must be defined relative to endpoints and timestamps. One-way delay generally requires synchronized clocks or a measurement arrangement that can reliably compare send and receive times. RTT is the elapsed time for a packet exchange to travel to a destination and for the corresponding response to return. TCP timing information or application request-response exchanges may provide RTT-related or response-time estimates, but these measurements can include endpoint processing and queueing. A single endpoint capture cannot generally isolate one-way network delay.

4.3 Packet loss

When corresponding transmitted and received packets can be matched, loss percentage may be calculated as $L = ((Ns - Nr)/Ns) \times 100$, where Ns is the number of packets sent and Nr is the number received over the same defined test interval. This calculation requires consistent observation points and reliable packet matching. A single capture does not necessarily provide both counts. A gap in sequence numbers or a TCP retransmission indicator is not, on its own, a complete end-to-end packet-loss measurement. RFC 2544 describes controlled frame-loss testing by comparing input and output counts under specified load conditions (Bradner & McQuaid, 1999).

4.4 Delay variation (jitter)

Jitter refers to variation in packet delay, but its calculation depends on the protocol and chosen definition. For real-time media, an RTP-specific estimate may be appropriate; for other traffic, a researcher may calculate variation in packet interarrival times. These quantities should not be conflated. A single capture can measure local interarrival timing but cannot independently recover true one-way network delay variation without suitable timing information.

4.5 Bandwidth utilization

A utilization estimate can be written as $U = (Ro/C) \times 100$, where Ro is the measured traffic rate and C is the relevant link capacity. An observed 40 Mbps on a 100 Mbps link corresponds to 40% utilization only if the rate and capacity are defined at compatible layers and refer to the same link. Nominal interface speed is not necessarily the bottleneck capacity of an end-to-end path.

4.6 TCP retransmissions

TCP retransmissions represent data sent again under TCP recovery mechanisms. A high count may be associated with performance degradation, but interpretation should consider total traffic, timing, duplicate acknowledgments, reordering, capture loss, and the capture location. A retransmission count is not automatically equivalent to a network-wide packet-loss rate.

5. Comparative analysis of methods

Table 2. Comparison of Wireshark analysis techniques and controlled benchmarking

Technique	Primary use	Key limitation
Protocol hierarchy	Characterize protocol distribution	Does not directly measure performance
Endpoint/conversation statistics	Identify high-volume hosts and exchanges	Limited to traffic visible in the capture
I/O graphs	Observe traffic variation over time	Observed rate is not maximum capacity
TCP analysis	Investigate transport events and retransmissions	Analyzer flags require contextual verification
Response-time statistics	Measure observable protocol exchanges	May include endpoint processing and both path directions
Controlled benchmarking	Measure performance under defined load	Requires suitable traffic generation and receiver-side measurement

The methods are complementary rather than interchangeable. Controlled benchmarking procedures for network interconnecting devices include throughput and frame-loss testing and emphasize repeatability and reporting (Bradner & McQuaid, 1999). RFC 6349 addresses sustained TCP throughput and explains why path characteristics and TCP behavior affect achievable performance (Constantine et al., 2011). Wireshark adds packet-level evidence during these tests but does not replace the test generator, receiver, or measurement protocol.

6. Proposed reproducible evaluation workflow

6.1 Test environment

A controlled laboratory can use two endpoint computers connected through a managed switch or a configured test network. One endpoint generates a predefined workload and the other measures successful reception. Record the topology, operating systems, interface types and speeds, link settings, Wireshark version, capture location, traffic-generation tool, workload parameters, and relevant offloading settings.

6.2 Workload design and capture

Workloads may include TCP file transfers, UDP streams, DNS queries, web requests, or application-specific exchanges. The workload, packet sizes, offered rate, duration, and number of concurrent flows should be chosen before capture. Where feasible, collect captures at both sender and receiver and retain traffic-generator logs. Use only authorized traffic and protect captures that may contain identifiers or sensitive content.

6.3 Analysis procedure

- Verify capture duration, interface, capture-drop statistics, and dataset completeness.
- Inspect protocol hierarchy, endpoints, and conversations to characterize the workload.
- Apply validated display filters to the relevant flows.
- Use I/O graphs and packet-length statistics to examine traffic patterns.
- Inspect TCP sequence behavior, retransmissions, and relevant protocol exchanges.
- Calculate each metric using an explicitly stated formula and measurement layer.
- Repeat each condition under comparable settings and report variation across trials.
- Document confounding factors, limitations, and the distinction between observation and inference.

6.4 Reliability and reporting

Each test condition should be repeated, with the number of trials stated. Report suitable summary statistics such as median, mean, standard deviation, and percentiles when appropriate. Consider background traffic, CPU load, storage activity, clock synchronization, interface offloading, and packet-capture drops as potential confounders. A complete report should state observation points, workload, duration, packet sizes, test settings, metric definitions, and the software versions used.

7. Discussion

Wireshark's principal strength is the combination of detailed protocol decoding, flexible filtering, flow-level summaries, and time-based visualization (Wireshark Foundation, n.d.). This combination can help explain behavior that aggregate interface statistics cannot reveal. For example, a reduction in application transfer rate can be investigated alongside TCP retransmissions, connection timing, packet-size distribution, and traffic-rate changes.

Several limitations remain. First, capture visibility is determined by the observation point. Second, encryption usually prevents inspection of plaintext application payloads, although endpoint, timing, packet-size, and transport metadata may remain observable. Third, metrics are definition-dependent: observed traffic rate, goodput, RTT, one-way latency, and protocol response time represent different quantities. Finally, capture loss and timestamp limitations can produce misleading interpretations. Reliable conclusions therefore require a controlled workload, a clear measurement definition, and corroborating evidence.

The strongest methodology combines Wireshark with complementary tools—for example, iPerf3 for active throughput testing, receiver-side counters for delivery confirmation, and system monitoring for CPU or interface conditions. Such measurements should be compared only when their measurement layers and observation points are compatible.

8. Research gaps and future directions

8.1 Automated traffic classification

Large capture files make manual inspection time-consuming. Future work could compare rule-based and machine-learning classification using documented, labeled datasets. Evaluation should report class balance, precision, recall, false-positive rates, and generalization to different network environments.

8.2 Encrypted traffic analysis

Encryption reduces payload visibility, motivating methods that use permitted metadata such as packet sizes, timing, flow duration, and directionality. Research should quantify classification accuracy and robustness while explicitly addressing privacy, consent, and dataset bias.

8.3 Anomaly detection

Potential systems could flag unusual traffic bursts, protocol distributions, connection patterns, or retransmission behavior. Evaluation should use labeled test cases and report detection delay, false alarms, computational overhead, and robustness to normal workload variation.

8.4 Reproducibility and cross-tool validation

Studies should specify topology, workload, capture point, duration, metric formulas, number of trials, and software versions. Results derived from packet capture should be validated against traffic-generator logs, receiver counters, or independent measurements where possible. RFC 2544 and RFC

6349 provide useful foundations for controlled network-device and TCP throughput testing (Bradner & McQuaid, 1999; Constantine et al., 2011).

8.5 Integrated monitoring

A future framework could combine packet-capture analysis with active probing, interface counters, and time-series monitoring. The main challenge is aligning timestamps and reconciling measurements collected at different protocol layers and observation points without treating them as equivalent.

9. Conclusion

This review has examined Wireshark as a tool for network traffic analysis and performance evaluation. Its capture, protocol dissection, filtering, endpoint and conversation statistics, I/O graphs, and TCP analysis features support detailed investigation of observed communication. However, packet inspection alone is insufficient to establish all end-to-end performance metrics. Throughput, goodput, RTT, one-way latency, packet loss, jitter, bandwidth utilization, and retransmissions must be defined and measured using methods appropriate to each quantity. Capture location, encryption, timestamp accuracy, packet drops, and background traffic can affect the validity of conclusions. A reproducible approach that combines Wireshark with controlled traffic generation, receiver-side measurements, repeated trials, and transparent reporting offers a more defensible basis for network evaluation. Further research should focus on automated analysis, encrypted-traffic characterization, anomaly detection, and cross-tool validation.

10. Limitations of the Review

This review is a narrative technical synthesis rather than a systematic review. The sources were selected for relevance to Wireshark packet analysis, measurement methodology, tool comparison, and network performance. No formal multi-database screening protocol or meta-analysis was performed. The included studies differ in objective, topology, workload, and measurement design; consequently, their numerical results are not pooled or treated as directly comparable. Future work should apply a reproducible database search and predefined inclusion criteria if a systematic-review claim is required.

Declarations

Funding: No external funding was received for this review.

Conflict of interest: The author declares no known competing financial interests or personal relationships that could have influenced the work reported in this paper.

Data availability: No new experimental dataset was generated or analyzed for this narrative review. The cited sources are publicly identifiable through the references.

REFERENCES

1. Alizan, A. N., & Zolkipli, M. F. (2024). A comparative study between Wireshark and Paessler Router Traffic Grapher (PRTG) in network monitoring and analysis. INTI Journal, 2024. https://doi.org/10.61453/INTIj.202451
2. Alwhbi, I. A., Zou, C. C., & Alharbi, R. N. (2024). Encrypted network traffic analysis and classification utilizing machine learning. Sensors, 24(11), Article 3509. https://doi.org/10.3390/s24113509
3. Birahmatika, R. A., Saedudin, R. R., & Kurniawan, M. T. (2024). Analisis perbandingan performansi jaringan wireless menggunakan software Wireshark dan Paessler PRTG di PT Industri Telekomunikasi Indonesia (Persero). eProceedings of Engineering, 11(4), 3721–3729. https://openlibrarypublications.telkomuniversity.ac.id/index.php/engineering/article/view/23698
4. Bradner, S., & McQuaid, J. (1999). Benchmarking methodology for network interconnect devices (RFC 2544). Internet Engineering Task Force. https://doi.org/10.17487/RFC2544
5. Constantine, B., Forget, G., Geib, R., & Schrage, R. (2011). Framework for TCP throughput testing (RFC 6349). Internet Engineering Task Force. https://doi.org/10.17487/RFC6349
6. Karamollahi, M., Williamson, C., & Arlitt, M. (2023). Packet-level analysis of Zoom performance anomalies. In Proceedings of the 2023 ACM/SPEC International Conference on Performance Engineering (pp. 221–232). Association for Computing Machinery. https://doi.org/10.1145/3578244.3583725
7. Shen, M., Ye, K., Liu, X., Zhu, L., Kang, J., Yu, S., Li, Q., & Xu, K. (2023). Machine learning-powered encrypted network traffic analysis: A comprehensive survey. IEEE Communications Surveys & Tutorials, 25(1), 791–824. https://doi.org/10.1109/COMST.2022.3208196
8. Sikos, L. F. (2020). Packet analysis for network forensics: A comprehensive survey. Forensic Science International: Digital Investigation, 32, Article 200892. https://doi.org/10.1016/j.fsidi.2019.200892
9. Soewito, B., Andy, Gunawan, F. E., & Mansuan, M. S. (2017). WAN optimization to speed up data transfer. Procedia Computer Science, 116, 45–53. https://doi.org/10.1016/j.procs.2017.10.007
10. Velan, P., Čermák, M., Čeleda, P., & Drašar, M. (2015). A survey of methods for encrypted traffic classification and analysis. International Journal of Network Management, 25(5), 355–374. https://doi.org/10.1002/nem.1901
11. Wireshark Foundation. (n.d.). Wireshark User's Guide. Retrieved October 28, 2026, from https://www.wireshark.org/docs/wsug_html/