IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received August 5, 2016, accepted September 5, 2016, date of publication September 14, 2016, date of current version October 6, 2016.

Digital Object Identifier 10.1109/ACCESS.2016.2609398

A Heterogeneous Service-Oriented Deep Packet Inspection and Analysis Framework for Traffic-Aware Network Management and Security Systems

MUHAMMAD ASRAR ASHRAF[1], HABIBULLAH JAMAL[2], SHOAB AHMED KHAN[3], ZAHEER AHMED[4], AND MUHAMMAD IRAM BAIG[1]

[1]Department of Computer Engineering, University of Engineering and Technology, Taxila 47080, Pakistan
[2]Ghulam Ishaq Khan Institute of Engineering Sciences and Technology, Topi 23640, Pakistan
[3]College of Electrical and Mechanical Engineering, Rawalpindi 46000, Pakistan
[4]Center for Advanced Studies in Engineering, Islamabad 44000, Pakistan

Corresponding author: M. A. Ashraf (asrarashraf@gmail.com)

This work was supported in part by the University of Engineering and Technology, Taxila, Pakistan and in part by the Center for Advanced Research in Engineering, Islamabad, Pakistan.

ABSTRACT A variety of Web-based applications, mobile apps, and other over the top data services with affordable 3G/4G enabled smart devices are major factors for enormous increase in heterogeneous data traffic at enterprise and mobile networks. This creates challenges regarding traffic management and requires traffic-aware intelligent network management to deliver sustained quality of experience for subscribers. Deep packet inspection and analysis (DPIA) provides base platform for development of traffic-aware intelligent network management and security systems. However, computationally complex DPIA-related packet processing for high speed data traffic makes these systems expensive. Furthermore, conventionally these traffic-aware network management and security systems are deployed in enterprise networks with independent and dedicated DPIA-related processing resources and require multiple copies of passively provisioned high speed data from network, while performing similar DPIA operations over the same data again and again. This duplicate deployment of expensive software and hardware resources for DPIA processing eventually results in higher capital expenditures as well as operational expenditures for network operators. We have proposed a novel service-oriented framework for heterogeneous deep packet inspection and analysis (SoDPI) that simultaneously provides diversified DPIA services to multiple client applications for network management and security operations in high-speed networks. Proposed framework provides flexible and comprehensive API-based service interface for client applications to register required DPIA services. SoDPI framework implementation is based on commodity hardware and deploys shared set of DPIA-related packet processing components, requiring only single copy of passive data provisioned from network. Experimental evaluations show that novel SoDPI framework requires considerably reduced amount of software and hardware resources to fulfill heterogeneous DPIA packet processing requirements for multiple client applications in comparison with conventional network management and security applications with dedicated DPIA components. This results in lower cost impacts for network operators with more network manageability.

INDEX TERMS Deep packet inspection, network monitoring, service oriented, traffic-aware network management, cost effective.

I. INTRODUCTION

Packet-data services have become dominant in communication networks with availability of numerous Layer-7 applications for interactive voice, video and data exchange along with continuous increase of available bandwidths for subscribers at cheaper prices. Network management operations have become progressively more challenging for service providers due to complexity of continuously

increased data traffic at network cores with reduced traffic visibility. This complexity becomes manifold with growing expectations of subscribers for higher quality of experience (QoE) with continuous emergence of new Layer-7 applications and higher network speeds. To meet these evolving challenges, it requires in-depth inspection of traffic by deploying traffic-aware systems for effective network management and to deliver superior end-user experiences [1]. Deep Packet Inspection and Analysis (DPIA) technology provides base platform for development of traffic-aware network management and security (T-NMS) systems. Layer-7 security firewalls, network intrusion detection systems, application-aware load-balancers and traffic management systems, data leakage prevention systems, application-aware QoE measurement systems, copyright enforcement systems and lawful interception systems are few examples of T-NMS systems [2]. Deployment of these traffic-aware systems ensure smooth network operations including security and management tasks in high speed enterprise networks for data service providers and ISPs, producing higher QoE for subscribers. However DPIA technology is computationally expensive that requires complex software and hardware components with high-end specifications to perform respective packet processing tasks resulting in higher capital expenditure (CAPEX) [3]. DPIA implementations also require to cater for continual increase in data rates at network cores along with addition of semantic analysis support for new Layer-7 applications and changes in existing Layer-7 applications. This makes DPIA technology further expensive due to higher operational expenditures (OPEX).

Prospective major components of a conventional DPIA packet processing setup as shown in Fig. 1, includes high speed packet capturing, protocol classification with pattern matching, data reassembly of TCP streams and Layer-7 protocol semantic analysis for contextual information extraction [4]. Each T-NMS system deploys a set of these DPIA components and features as per requirements to perform respective system operations.

T-NMS systems that are deployed for multiple network security operations require comprehensive set of DPIA components mentioned in Fig. 1. For example, application-aware Network Intrusion Detection/Prevention Systems (NIDS/NIPS) perform application-aware pattern matching at packet payloads after Layer-7 protocol specific semantic analysis to identify malicious contents and perform mitigation of intruders and infected entities in any network [5]. Data Leakage Prevention Systems (DLPS) [6] and Copyright Protection Systems [7] also require extensive DPIA processing to extract Layer-7 application specific fields and spot confidential and copyright contents being illegally transferred over network by using email, file transfer, media sharing and Point-2-Point (P2P) applications. Lawful Interception (LI) systems [8] are deployed as regulatory and security requirements to monitor malicious activities of cyber terrorists and other miscreants involved in illegal and security threating acts. LI systems require DPIA processing components including real-time packet capturing, protocol classifications, and stream data reassembly with Layer-7 protocol parsing to extract information and details about cyber activities of miscreants and terrorists.

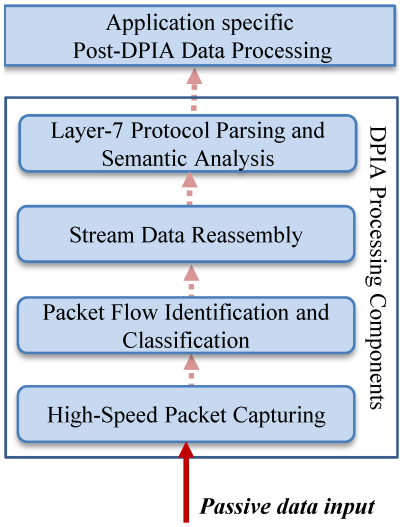

FIGURE 1. Prospective components for DPIA processing.

On the other side, T-NMS systems that are deployed for traffic-aware network management operations mostly require a subset of DPIA packet processing components to perform T-NMS application specific data processing. For example, network traffic management and planning applications require high-speed packet capturing and protocol classification for identification of Layer-7 application types and compute application-aware traffic statistics including payload sizes, packet counts and flow counts for traffic management and planning network expansion [9], [10]. QoE and network performance measurement applications require packet protocol classification to collect application-aware network performance statistics for estimation of QoE metrics. This helps network administrators to identify low performing areas and nodes in network and to take necessary corrective measures [11]. Similarly application-aware policy and charging control (PCC) systems require application types and respective payload sizes of data packets to perform application-aware charging for multiple data services [12].

Any enterprise network requires multiple DPIA based T-NMS systems for respective network operations, security and management tasks. Cost impact of expensive DPIA processing components becomes manifold when multiple T-NMS systems with dedicated DPIA resources are deployed in same network as per conventional approach depicted in Fig. 2.

Each T-NMS system equipped with independent DPIA related resources results in repeated processing of same data over and over for similar set of DPIA functions. It requires more number of complex packet processing software and hardware components, associated logistic resources

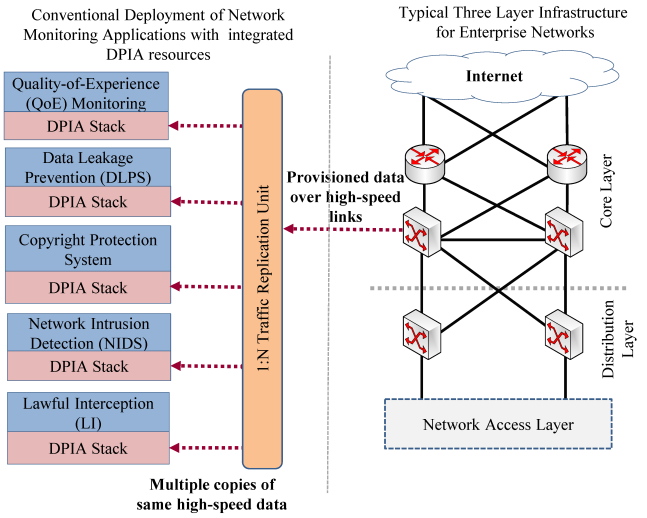

FIGURE 2. Conventional deployment approach for T-NMS systems.

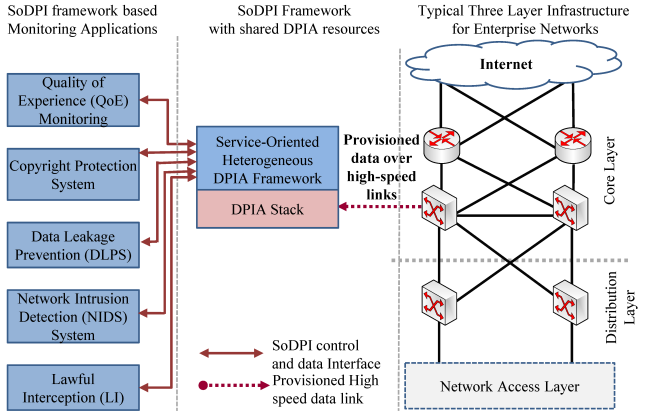

FIGURE 3. SoDPI framework based deployment of T-NMS systems.

(e.g. energy consumption, space etc.) and multiple high speed passive data provisioning links with respective high speed traffic replication units for these systems to perform similar data processing. This ultimately results in increased CAPEX as well as high OPEX for network operators due to increase in required network management activities and continuous system up-gradations.

In light of the issues outlined above, we introduce the Service- Oriented Heterogeneous Deep Packet Inspection and Analysis Framework (SoDPI), a passive network monitoring and analysis framework with shared deployment of DPIA processing resources while simultaneously providing multiple DPIA services to diversified T-NMS clients. SoDPI framework provides abstraction for incorporated DPIA packet processing services via a set of flexible API functions. Multiple T-NMS client applications can register for required DPIA services via these API functions. Proposed deployment of T-NMS systems based on novel SoDPI framework is shown in Fig. 3. The outcomes of the proposed SoDPI framework are (i) Simultaneous provisioning of heterogeneous DPIA services for multiple T-NMS client systems (ii) Reduced amount of required hardware, software and associated resources for computationally expensive DPIA processing (iii) Reduced CAPEX and OPEX for service providers in comparison to conventional approach for deployment of T-NMS systems.

We have evaluated SoDPI framework in a 10GE network environment with a large data set of real packets traffic. Evaluations are performed with two example monitoring applications, each with different DPIA processing requirements (i) Protocol-Aware traffic statistics collector application to collect and display Layer-7 protocol based traffic statistics (ii) Data transfer monitoring application to identify data transferred via SMTP [34] and POP3 [35] protocols. Both example applications are evaluated with SoDPI framework based approach and conventional approach. Evaluation results show that with same accuracy performance, proposed SoDPI framework requires considerably reduced data processing and logistical resources for heterogeneous DPIA packet processing in comparison to conventional DPIA based monitoring applications.

The rest of this paper is organized as follows. We start by reviewing existing related work in section 2. Before describing detailed architecture, we discuss design considerations of proposed SoDPI framework with respective features in section 3. Section 4 explains architecture of proposed framework. Section 5 provides estimation and comparison of required processing resources for conventional approach and SoDPI framework based monitoring applications. Section 6 contains implementation details for two example applications based on SoDPI framework. Experimental evaluation of proposed framework along with comparisons of results is discussed in section 7, while section 8 concludes the paper.

II. RELATED WORK

Extensive research work has been done related to complex packet processing for DPIA operations in high speed networks. Major areas focused include high-speed packet capturing implementations with zero packet drops [15]–[19], improvements in throughputs of pattern matching algorithms [2], accurate packet classifications algorithms [21]–[25], optimal stream reassembly implementations [26]–[28] and extensive Layer-7 protocol semantic analysis [5], [29], [30]. Another key area for DPIA related research consists of DPIA implementations based on commodity hardware based platforms instead of dedicated FPGA and network processor units to reduce cost impacts [13], [14]. However there has been very little work done related to shared deployment of DPIA processing components to reduce cost impacts due to duplicate deployment of packet processing resources except few primitive implementations as suggested in [32] and [33].

Moreno in [32] has proposed a monitoring platform termed as multi-granular, multipurpose and multi-Gb/s monitoring (M3Omon) that acts as an intermediate packet capturing layer and serves client applications for packet data along with basic packet level statistics. However, it does not address

common key features of DPIA processing including pattern matching, application protocol classification and semantic analysis for extraction of Layer-7 application specific features as required for traffic-aware network operations, security and management systems.

Similarly Bremler in [33] has proposed an inline DPIA framework to provide centralized pattern matching service for different network elements. It proposes a single DPIA controller module with multiple distributed DPIA modules deployed in different network segments. It only operates in chained inline mode with focus on raw pattern matching of packet contents. It does not address DPIA requirements for application-aware network management and security tasks that include protocol classification, stream data reassembly and semantic analysis for Layer-7 protocols. It does not provide aggregated packet-level and application-level flow statistics that are basic requirements for traffic-aware network management systems. Another important feature required to deploy DPIA as shared service is extensive and flexible communication framework that enables smooth access of heterogeneous DPIA services by multiple client applications. This important feature is also not addressed in proposed implementation.

Our studies in related works found that no study exists for a shared and cost-effective DPIA based network traffic monitoring and analysis framework that can fulfill diversified DPIA processing requirements of multiple T-NMS systems related to traffic-aware network operations, management and security functions in high-speed networks.

III. DESIGN CONSIDERATIONS FOR PROPOSED SODPI FRAMEWORK

Key design considerations for proposed SoDPI framework with respective benefits are as following:

- *Service Oriented architecture* based DPIA processing approach ensures simultaneous provisioning of DPIA services to multiple clients while performing single time DPIA related packet processing of high speed data in comparison to conventional deployment approach. This results in reduced CAPEX for service providers. DPIA processing components also require frequent updates in terms of hardware and software to keep monitoring functions equally effective. Enterprise networks keep on adding new high-speed data links at backbones to cater high bandwidth demands. Similarly new Layer-7 applications and smartphone apps keep on being introduced that require addition packet classification and information extraction functions for of respective Layer-7 protocols. With conventional design approach, required hardware and software updates are performed at all deployed T-NMS applications separately. However with SoDPI framework, required updates and addition of software and hardware components are performed at single SoDPI framework instance. This simplified network management results in reduced operational and maintenance costs as well for network operators.
- *Heterogeneous DPIA services* are simultaneously provided to multiple client applications from single DPIA setup deployment. DPIA services offered by proposed framework include:
  - General packet traffic statistics
  - Traffic statistics based on Layer-7 protocols.
  - Reassembled data access of TCP streams
  - Access of extracted protocol fields and contents after semantic analysis of L-7 protocols.
- *Abstraction for all DPIA related complex packet processing* enables researchers and application developers to focus on development of application specific features of T-NMS systems. They don't require in-depth knowledge of complex DPIA processing functions and algorithms, ultimately resulting in rapid development of different T-NMS system applications.
- *Use of existing implementations for core DPIA processing* saves efforts that are required to design and develop core DPIA components and enables to focus on improvements and feature additions of T-NMS applications.
- *Commodity off-the-shelf Hardware* based approach for SoDPI architecture provides flexibility and scalability as well as lower costs in comparison to legacy FPGA and ASIC based expensive DPIA systems that are not easy to scale [13], [14].

IV. ARCHITECTURE

As per given architecture for the proposed SoDPI framework in Fig. 4, client applications avail required DPIA services by making service requests via defined API functions calls. SoDPI framework performs DPIA related packet processing of high speed network data according to client requests and forwards DPIA processing results to respective clients in the form of pre-defined metadata formats along with contents.

Architecture of proposed framework is stratified into three layers with each layer encapsulating different functionalities. Functions for service management, client interfacing and dispatching of DPIA processing results are performed by Service Management and Data Dispatching (SMD) layer while DPIA based packet processing and related service abstraction functions required for DPIA services are provided by Core DPIA Processing (CDP) layer and DPIA Services Abstraction (DSA) layer respectively.

To elaborate details of proposed SoDPI framework, firstly we discuss service management and client interfacing functions at SMD layer. Thereafter we elaborate composition and respective data processing flows for DPIA services at CDP and DSA layers respectively. In the last part of this section, we discuss data dispatching of DPIA processing results to respective client applications.

A. SERVICE MANAGEMENT AND CLIENT INTERFACING

Functions for the management of DPIA service requests from clients are primarily provided by the *Service Management*

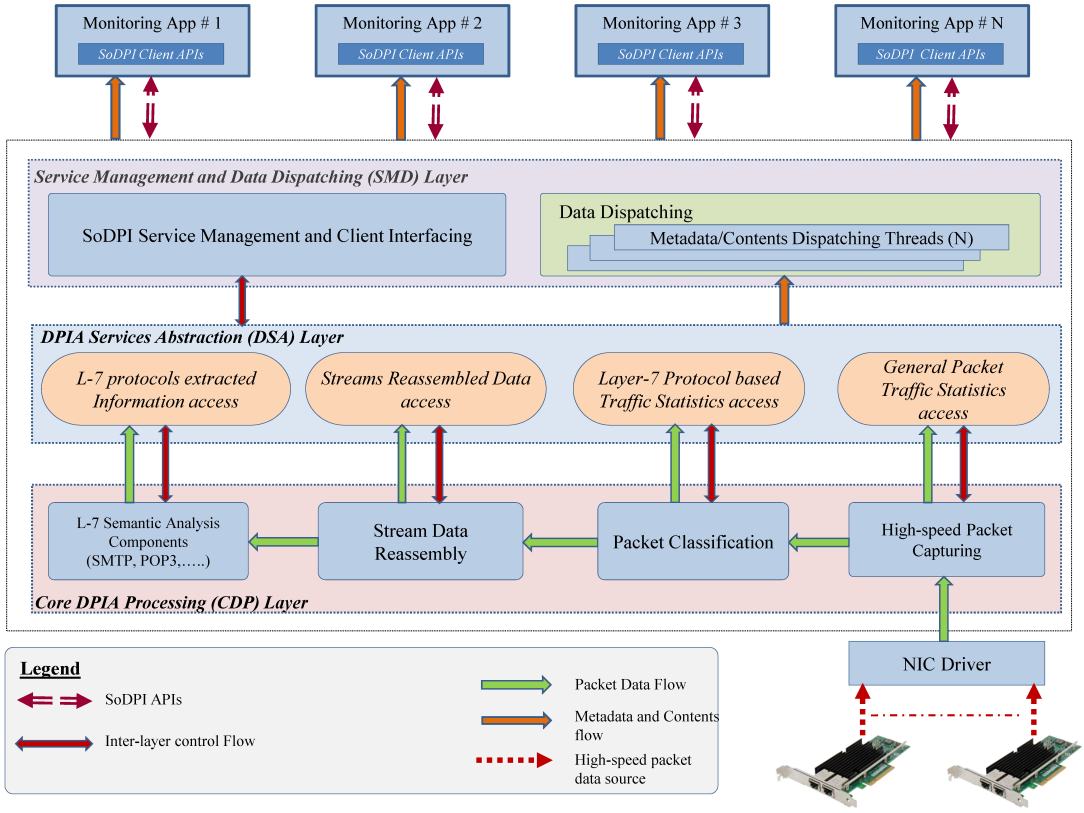

FIGURE 4. SoDPI framework based deployment of T-NMS systems.

and Client Interfacing component at SMD layer that acts as immediate interface for client applications to avail DPIA services offered by proposed framework. Service management functions include establishment of service sessions with multiple clients, receiving requests for DPIA services from clients, interpreting service requests, forwarding to core DPIA packet processing and service abstraction components of respective DPIA services and transmission of responses to client applications against respective service requests.

Key functional units of *Service Management and Client Interfacing* component are depicted in Fig. 5.

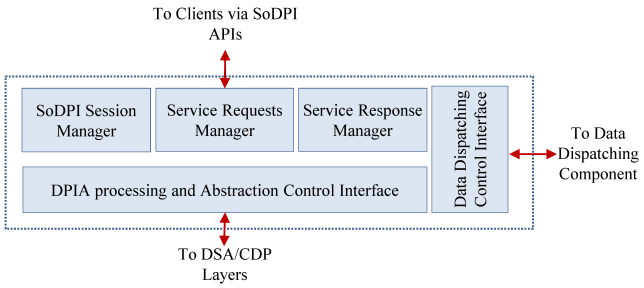

FIGURE 5. Functional blocks of service management and client interfacing component.

All requests from clients related to DPIA services are received and processed by *Service Request Manager* that segregates the requests based on respective types and perform further processing accordingly. Proposed SoDPI framework offers comprehensive set of API functions as listed in Table 1, for clients to avail required DPIA services.

List of SoDPI framework APIs consist of function calls for initialization, configuration, registration and termination of DPIA service sessions. To elaborate service access procedures and respective request management and processing details, we present a scenario with an example client application that registers for *general packet traffic statistics service*. Sequence of API function calls made by client application and respective request processing by *Service Management and Client Interfacing* component is depicted in Fig. 6.

As per given service access procedure, new service session requests from clients are processed by *SoDPI Session manager* that assigns unique *Session Ids* to respective clients along with maintaining client details. After session initialization, client applications make configuration API function calls to specify storage location for dispatching of DPIA processing results to respective clients.

Clients make requests for DPIA services via respective service registration APIs after successful service configuration. All service requests from connected clients are assigned unique *Request Ids* within the scope of assigned *Session Ids* for respective clients. Assigned *Session Ids* and *Request Ids* are key identification elements for all subsequent

TABLE 1. Main functions of SoDPI APIs.

Session Initialization, Configuration and Termination APIs	
int32 ***sodpi_create_session(***unit32 SoDPI_server_IP ,uint16 SoDPI_server_port***)***	Creates a DPIA session with SoDPI Server network address details and returns with *SoDPI Session Id.*
int32 ***sodpi_start(***const int32 sodpi***)***	Starts DPIA processing for registered service for respective client with given *SoDPI Session* configuration and returns with respective status as success or failure.
int32 ***sodpi_stop(***cont int32 sodpi***)***	Stops DPIA processing for registered service operation of respective *SoDPI Session*
int32 ***sodpi_close_session(***const int32 sodpi***)***	Terminates client session and deallocates all resources allocated for respective *SoDPI Session.*
int32 ***set_transfer_location_for_DPIA_results(***const int32 sodpi, uint32 serverIP, char * meta_dir_path, char * username, char * password***)***	Sets client side FTP service credentials for transfer of DPIA results from SoDPI service instance to client.
General Packet Statistics Service API	
int32 ***register_pkt_stats_access_for_IPs(***const int32 sodpi, uint32 start_ip,uint32 end_ip, uint16 capture_interval***)***	Provides packet level traffic statistics for given IP range periodically after set time interval value in seconds. Returns with respective *Request Id* in case of success or with error code in case of failures. Same valid for all other DPIA service registrations APIS.
Layer-7 Protocol Traffic Statistics Service API	
int32 ***register_aggregated_stats_access_for_Protocols(***const int32 sodpi, uint32 start_ip,uint32 end_ip,uint capture_interval, struct list_of_protocols * protos ***)***	Provides aggregated traffic statistics for given list of protocols with specified IP range periodically after set time interval value
Service API for Stream Reassembled Data Access	
int32 ***register_stream_reassembled_data_access(***const int32 sodpi, uint32 start_ip, uint32 end_ip, uint capture_interval, struct list_of_protocols * protos, eDirection dir, int cuttoff_stream_size, ePortType port_type, eLevelOfAccess access_level***)***	Provides stream re-assembled data for given list of protocols with specified IP range and payload cutoff size for reassembled at given time intervals with required level of services access
Service APIs for Layer-7 Protocols Extracted Information Access API	
int32 ***register_layer7_parsed_data_access (***const int32 sodpi, uint32 start_ip,uint32 end_ip, struct list_of_protocols * protos, eLevelOfAccess access_level***)***	Provides layer 7 parsed data with extracted metadata and contents for respective protocol list and specified IP range with required level of services access

communication and DPIA processing during service sessions with clients. DPIA service requests along with *Session Ids*, *Request Ids* and input filtering parameters are then forwarded to packet processing and abstraction components of respective DPIA services at CDP and DSA layers via *DPIA Processing and Abstraction control interface.* Filtering parameters are utilized by DPIA services to process and filter packets accordingly. *Service Response Manager* manages all responses against initialization, configurations and DPIA service API functions calls. It forwards these responses to client applications as status of respective DPIA service requests. Responses consist of information regarding success or failure against service requests with respective error details.

B. DATA PROCESSING ARCHITECTURE FOR DPIA SERVICES

Once DPIA service sessions are successfully established with respective clients , packet processing of high speed input data starts at at CDP and DSA layers. Overall data processing flow for DPIA services with inter-service and intra-service data exchange details are illustrated in Fig. 7. As per given sequence, each DPIA service provide respective packet processing functionalities to client applications as well as to successor DPIA services. For example, data packets collected by *Packet Capturing* component are required for protocol classification, L-7 protocol identified data from *Protocol Classification* component is required for protocol-level stream data reassembly and stream reassembled data from *Stream Data Reassembly* component is required for L-7 semantic analysis and extraction of protocols fields and contents.

To elaborate compositions and functions of multiple DPIA services, we first discuss generic composition and data processing architecture of a DPIA service. Later we discuss specific details of all DPIA services with respective components specifications, data processing flows and results.

1) GENERIC ARCHITECTURE AND COMPOSITION OF DPIA SERVICES

Each DPIA service consists of two components as shown in Fig. 8, with a core DPIA packet processing component at

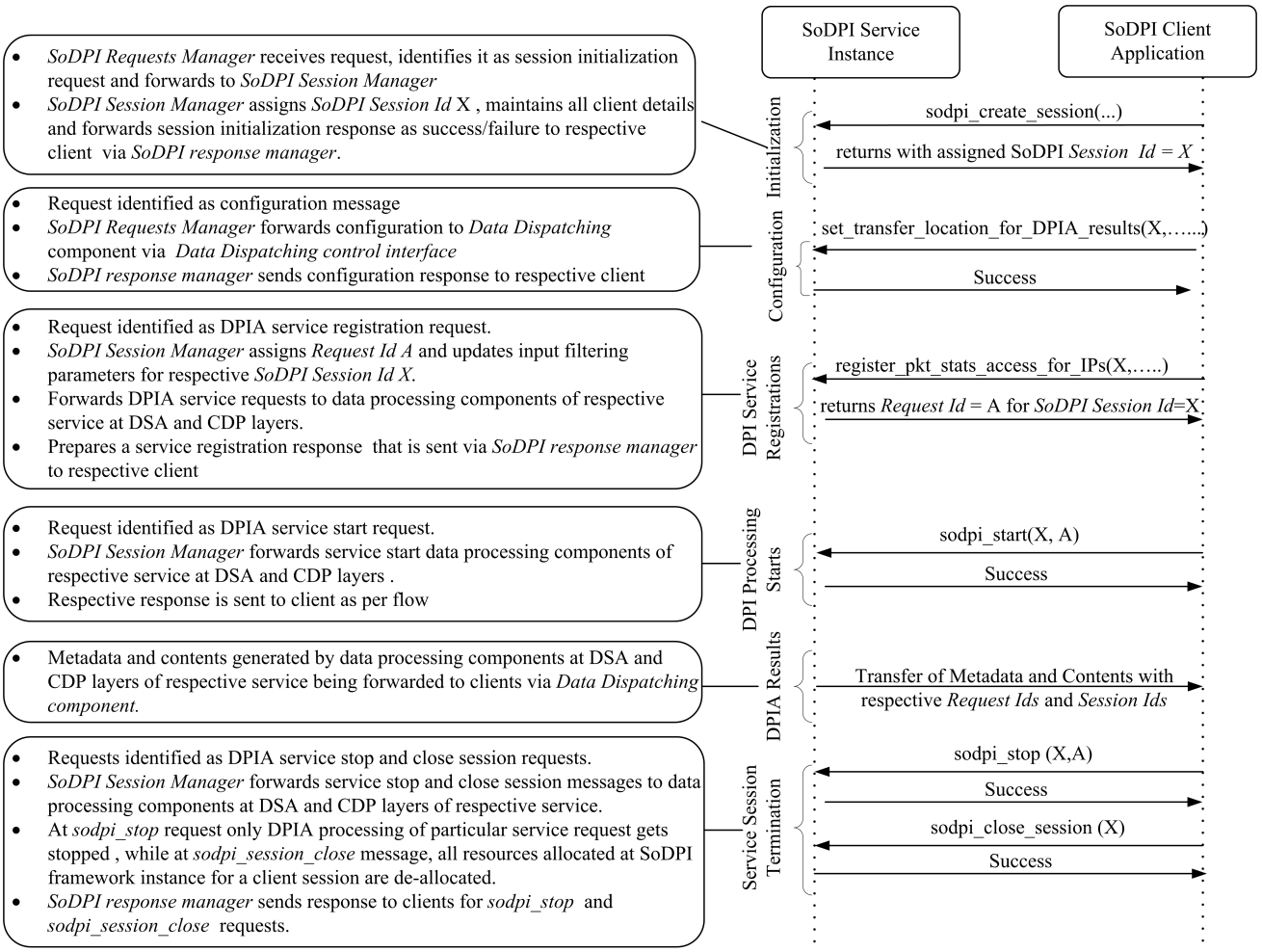

FIGURE 6. Sequence of SoDPI framework API calls for DPIA services access with processing details.

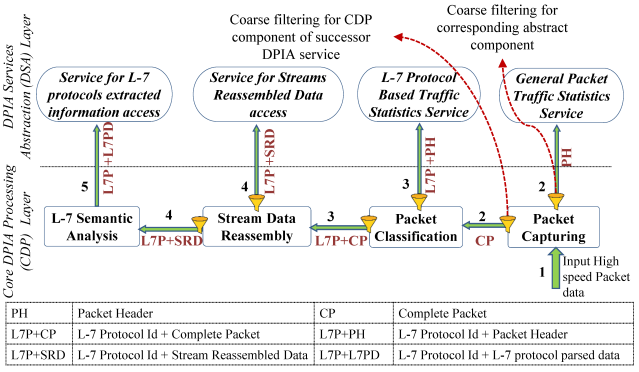

FIGURE 7. DPIA related packet processing flow at CPD and DSA layers.

CDP layer and a corresponding service abstraction component at DSA layer.

Core DPIA Packet Processing component of a DPIA service at CDP layer receives input packet data from its predecessor components or NICs as per given data processing sequence in Fig. 7. Data processing flow for *Core DPIA Packet Processing* component with functional details is depicted in Fig. 8. It performs two major functions i.e. (i) Service specific core DPIA processing and (ii) Coarse filtering of processed data. As per SoDPI framework design, existing state of the art implementations are proposed for core DPIA related processing functions, with details discussed in coming sub-sections for respective DPIA services. After completion of component specific core DPIA processing, it becomes processing intensive and resource consuming to pass all processed data to respective service abstraction components at DSA layer and to core DPIA components of successor DPIA services. To optimize resource consumption, processed data is passed to coarse filtering block that maintains filters for its corresponding abstraction component at DSA layer and for its successor DPIA services. With coarse filtering, only required traffic data is forwarded to these components, while rest of traffic data is discarded. Coarse filtering for successor DPIA service helps to reduce amount of data processing by these DPIA services, while coarse filtering for corresponding service abstraction component at DSA layer ensures forwarding of only required DPIA processed data for

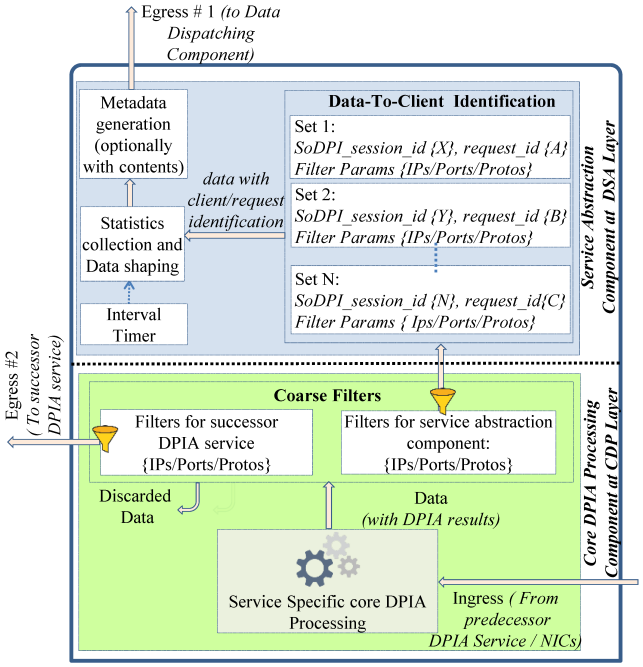

FIGURE 8. Generic architecture and data processing flow for DPIA services.

respective shaping and forwarding to clients. Coarse Filters consist of IP addresses, list of L-7 protocols and TCP/UDP ports.

Service abstraction component of a DPIA service receives DPIA processed data from corresponding *core DPIA processing component* and performs three major functions i.e. (i) Data-to-client identification (ii) Statistics collection and data shaping (iii) Metadata generation.

As shown in Fig. 8, after receiving ingress packets from *core DPIA processing component*, first function performed is *Data-to-client identification* to identify clients that have requested for respective DPIA processed data. This processing block maintains record sets with respective *Session Ids*, *Requests Ids* and filtering parameters that are received from respective clients as input parameters of SoDPI API function calls. Filtering parameters vary for different DPIA services and can include IP addresses, port addresses and list of Layer-7 protocols as requested by T-NMS clients. *Data-to-client identification* process finds *Session Ids* along with *Request Ids* for DPIA processed data based on these filtering parameters. After *Data-to-client identification* process, data is labeled with respective *Session Ids* and *Request Ids* for identification during further data processing and dispatching of results to clients.

Next function is *statistics collection and data shaping* as per functional requirements of each DPIA service. For abstraction components of DPIA services including general packet traffic statistics and Layer-7 protocol based traffic statistics, only respective traffic statistics are maintained while for other DPIA services, statistics along with original packet payload data is also maintained if required by clients in service specific forms. After statistics collections and data shaping, *metadata generation* block performs generation of service specific metadata objects. Pre-defined metadata formats are proposed for results of each DPIA service that are elaborated in coming sub-sections with respective details for these services. Metadata objects for SoDPI clients are identified by assigned *Session Ids* and *Request Ids* and *Transaction Ids*. *Transaction Ids* are used for identification of multiple metadata objects generated against same *Requests Ids* and *Session Ids*. Metadata generation takes place either at pre-defined time intervals or at captured L-7 protocol specific events. Pre-defined time intervals with time resolution in seconds, can be configured by client applications at DPIA service registration time via respective APIs. Generated metadata objects by each DPIA service are passed to *Data Dispatching* component at SMD layer that forwards these metadata objects to respective clients.

2) GENERAL PACKET TRAFFIC STATISTICS SERVIC

Multiple traffic engineering applications require general traffic statistics including aggregated as well as direction-wise packet statistics for up-streams and down-streams to perform network performance measurements. This helps administrators to identify exhaustion of high speed data links, frequency of connection terminations, fluctuations in core network bandwidth, events of traffic bursts and monitoring of respective effects along with preparation of daily traffic graphs.

T-NMS client applications make access requests for general packet traffic statistics service via *register_pkt_stats_access_for_IPs ()* API call as given in Table 1, with respective input filtering parameters.

As per SoDPI architecture in Fig. 4, this service deploys *Packet Capturing* as core DPIA processing component at CPD layer and *General Packet traffic statistics access* as service abstraction component at DSA layer.

Packet Capturing is the first CDP layer component as per DPIA processing sequence given in Fig. 7. It performs real-time capturing of high-speed packets from NICs. Key requirements for any packet capturing implementation are real-time packet access capabilities at high speed data rates without packet drops. Conventionally, it required the use of specialized hardware based on ASICs, reprogrammable FPGAs or network processors to ensure high-speed zero-drop packet capturing. These solutions offer high-throughputs with real-time performance but lack flexibility and scalability. As an alternative, Rizzo et al. [13] and Garcia-Dorado et al. [14] have proposed the utilization of commodity hardware based packet capturing solutions that provide flexibility and scalability with reduced capital and operational expenditures. Multiple state of the are packet capturing implementations including PF_RING DNA [15], PF_RING ZC [16], netmap [17], PFQ [18] and Intel DPDK [19] exist that are developed by incorporating NUMA based multi-processor architecture with customized NIC drivers to achieve capturing throughput data rates up to multi-10Gb/s. We have interfaced PF_RING DNA [15]

implementation as core packet capturing unit for proposed SoDPI framework. It offers extensive set of APIs and provides capturing speeds ranging from 1Gb/s to 40Gb/s.

As per DPIA service architecture shown in Fig. 8, *packet capturing* component also performs coarse filtering of captured data and forwards packets with payloads to successor DPIA service component of *protocol classification* as complete packet data is required for protocol classification of most Layer-7 protocols. It also performs coarse filtering and forwarding of only packet headers to its corresponding service abstraction component of *general packet traffic statistics access* at DSA layer to save memory resources by avoiding unnecessary copying of complete packet data as only packet headers are required for calculation of general packet traffic statistics.

General packet traffic statistics access component at DSA layer provides service abstraction for *packet capturing* component at CDP layer. It computes required packet level statistics including byte counts and packet counts against given input filtering parameters provided by respective clients. Separate statistics counters are maintained for outbound and inbound data traffic. Packet traffic statistics are periodically forwarded to *metadata generation* block at requested time intervals that generates metadata objects as per *struct* ***PktStatsMeta*** and forwards to data dispatching component at SMD layer. Format for metadata fields with respective details for *General Packet traffic statistics* service are given in Table 2. Separate metadata objects are generated in case of statistics with similar filtering criteria requested by multiple clients.

TABLE 2. Metadata fields for general packet traffic statistics service.

Fields		Description
`struct`	**`Base_Metadata`** {	
`int32`	**`SoDPI_session_id`**;	Session Id assigned by SoDPI framework
`int32`	**`request_id`**;	Request Id for registered DPIA service
`uint64`	**`transaction_id;};`**	ID for each metadata transaction with in scope of respective request_id and sodpi_session_id
`struct`	**`PktStatsMeta`**{	General Packet Traffic Statistics
`struct`	`Base_Metadata` **`base_parent_ids`**;	Set of IDs against registered Service as per struct Base_Metadata
`struct`	`timeval` **`start_time`**, **`end_time`**;	Start and end time of capture
`uint64`	**`in_Bytes_count`**, **`out_Bytes_count`**;	Byte level statistics for up and down streams
`uint64`	**`in_PktCount`**, **`out_PktCount`**;	Packet level statistics for up and down streams

3) LAYER-7 PROTOCOLS BASED TRAFFIC STATISTICS SERVICE

Traffic management systems with Layer-7 application-awareness enable network administrators to manage application-level QoS with guarantees as per service agreements and reduce the chances of traffic congestions for bandwidth hungry and delay sensitive applications for voice and video communication. Similarly application-aware traffic statistics identify bursts in traffic loads at network cores caused by rogue applications that exploit network misconfigurations and results in overall network degradations. Application-awareness in traffic measurements tools provides capability to rate limit or drop traffic of these rogue applications.

Layer-7 Protocols based Traffic Statistics Service at SoDPI framework provides Layer-7 protocol based traffic statistics to T-NMS applications for traffic-aware network management, application-based QoS management and application-aware charging systems. T-NMS clients request for Layer-7 protocol based traffic statistics service via *register_aggregated_stats_access_for_Protocols ()* API calls as per Table 1.

Implementation of this service incorporates *Protocol Classification* component as core DPIA processing component at CPD layer and *Layer-7 Protocol based traffic statistics access* as service abstraction component at DSA layer.

Protocol classification component performs identification of Layer-7 protocols for high speed input packet data received from *packet capturing* component as per Fig. 7. Commonly used protocol classification methodologies include port numbers, payload inspection, packet statistics and machine learning algorithms (MLAs) based techniques with respective accuracy performance [20]. *Protocol classification* implementations require continual updating due to frequent changes of Layer-7 application protocols and arrival of new Layer-7 applications. Many protocol classification implementations including PACE [21], nDPI [22], Libprotoident [23] and L7filter [24] are developed with respective features and shortcomings [25]. For SoDPI framework, nDPI [22] is incorporated for core packet classification functions that provide classification support for a large set of Layer-7 applications with frequently available updates.

Protocol classified data is forwarded to corresponding service abstraction component of *Layer-7 protocol based traffic statistics access* at DSA layer and to *Stream Reassembly* component at successor DPIA service after respective coarse filtering as shown in generic architecture and data processing flow for DPIA services in Fig. 8.

Layer-7 protocol based traffic statistics access component computes traffic statistics including L-7 protocol based flow counts, packet counts and bytes counts for both outbound and inbound traffic as per filtering criteria from connected clients. These statistics are forwarded to *metadata generation*

block at requested time intervals. Metadata generation for protocol classified statistics is performed as per *struct* ***ProtoLevelStatsMeta*** with respective data fields and details given in Table 3.

TABLE 3. Metadata fields for Layer-7 protocol based traffic statistics service.

Fields		Description
struct	**ProtoLevelStatsMeta**{	
struct	Base_Metadata **base_parent_ids**;	Set of IDs against registered Service as per struct Base_Metadata
struct	timeval **start_time**, **end_time**;	Start and end time of capture
struct	**proto_stats {**	Protocol wise statistics
	eProtocols **proto**;	L-7 protocol id
	uint64 **in_Bytes**, **out_Bytes**,	Byte level statistics for up and down streams
	uint64 **in_Packets** **out_Packets**;	Packet level statistics for up and down streams
	uint64 **FlowCount**;	Total flows for given protocol
	proto_stats ***next**;	Next protocol Statistics
	}**proto_stats_obj**;};	

4) SERVICE FOR STREAMS REASSEMBLED DATA ACCESS

Stream reassembly is another feature of DPIA technology required for applications like lawful interception systems, network intrusion detection and prevention systems and other network security centric monitoring systems that require stream reassembles data to perform comprehensive protocol semantic analysis and respective monitoring functions.

Service for Streams Reassembled Data access provides reassembled data of TCP streams to client applications as per requirements. Clients make requests for stream reassembled data via *register_stream_reassembled_data_access ()* API call with respective input parameters.

Service consists of *Stream Reassembly* as core DPIA processing component at CDP layer and *Streams Reassembled Data access* as service abstraction component at DSA layer.

Key challenge for *Stream Reassembly* implementations is optimal memory management to handle large number of streams with variable size of reassembled data. Multiple stream reassembly implementations exist including SCAP [26], Snort [27] and Libnids [28] that provide reassembled data for TCP streams of passively provisioned data. We have incorporated Libnids [28] for stream data reassembly functions for evaluation of proposed SoDPI framework. After data reassembly of required TCP streams, coarse filtering functions are performed and filtered data is forwarded to corresponding abstraction component of *Stream Reassembled Data access* at DSA layer and to *L-7 protocol semantic analysis* component of successor DPIA service with respective coarse filtering criteria.

After receiving reassembled data from *Stream Data reassembly* component, *Streams Reassembled data access* component provides stream statistics along with reassembled data, depending upon type of access requested from respective clients. Clients requiring metadata access are provided only required stream statistics while client requesting metadata with contents access are provided reassembled data as well. Statistics along with reassembled data are passed to *metadata generation* block either on configured time intervals by clients or on occurring of termination events for TCP connections including FIN and RST events. *struct* ***StreamReassemblyMeta*** is used as format for generation of metadata objects for DPIA results of this service. Respective data fields and details of *struct* ***StreamReassemblyMeta*** are given in Table 4.

TABLE 4. Metadata fields for streams reassembled data access service.

Fields		Description
Struct	**StreamReassemblyMeta**{	
struct	Base_Metadata **base_parent_ids**;	Set of IDs against registered Service as per struct Base_Metadata
struct	timeval **start_time**, **end_time**;	Start and end time of capture
uint32	**client_ip**, **server_ip**;	Client and server IP of stream
uint16	**source_port**, **dest_port**;	Source and destination port
uint32	**stream_byte_count**, **stream_pkt_count**;	Data statistics counters of a stream data
eDirection	**dir**;	Direction of stream
eProtocols	**proto**;	L-7 application protocol for stream
uint32	**len_of_contents**;	Length of contents buffer
char *	**contents**;};	L-7 protocol data contents

5) SERVICE FOR L-7 PROTOCOLS EXTRACTED INFORMATION ACCESS

Reverse engineering of Layer-7 protocol and extraction of respective protocol fields along with contents are essential for multiple network monitoring and security applications including Lawful Interception (LI) systems and many other network forensics applications.

Access of L-7 protocol fields and contents after semantic analysis is provided by *Service for L-7 protocols extracted information access*. Clients requiring L-7 protocol extracted information make service requests via *register_layer7_parsed_data_access ()* API calls with specified

list of L-7 protocols and other input parameters. This service consists of *L-7 Semantic Analysis* component as core DPIA processing component at CDP layer and *L-7 protocols extracted information access* as service abstraction component at DSA layer.

L-7 Protocol Semantic Analysis performs regular expression based extensive pattern matching to parse reassembled data for multiple L-7 protocols. *L-7 protocol semantic analysis* component can consist of multiple protocol analyzers that perform extraction of information including data fields, content types and transferred contents from data of respective L-7 protocols. FlowSifter [29] and ultra-PAC [30] are few examples of parsing and semantic analysis platforms to develop Layer-7 protocol analyzers. There is a large set of Layer-7 applications including voice, video and data applications that require development of respective protocol analyzers for semantic analysis. We have implemented custom L-7 parsing and semantic analyzers for email protocols including SMTP [34] and POP3 [35] for evaluation of SoDPI framework. Protocol analyzers parse packets according to semantics of SMTP and POP3 protocols to extract details of data fields and details of protocol specific events.

L-7 protocol parsed data is forwarded to corresponding abstraction component of *L-7 protocols extracted information access* for further processing and metadata generation. Service abstraction component provides DPIA results in the form of metadata only or metadata with contents. For this specific service, metadata generation takes place solely based on L-7 protocol specific events instead of interval timer. Example of protocol specific events includes session start, message transfer, file transfer, session termination and other L-7 protocol specific events. Metadata generation format used by this abstraction component is as per *struct L 7_Proto_Parsed_Meta* with respective fields and details given in Table 5.

C. DISPATCHING OF DPIA PROCESSED DATA

After core DPIA processing, data shaping and metadata generation functions for all DPIA services at CDP layer and DSA layer respectively, generated metadata objects are forwarded to *Data Dispatching component* at SMD layer that dispatches these results to respective client applications based on *Session Ids*. DPIA processing results of each service consists of respective pre-defined metadata and contents. Metadata with contents are stored in the form of data files temporarily and dispatched to respective clients via FTP service. In order to have near real-time and parallel delivery of metadata objects, *Data Dispatching component* consists of dedicated dispatching threads for each connected T-NMS client.

V. ESTIMATION OF REQUIRED RESOURCES FOR SODPI FRAMEWORK

For estimation and comparison of required resources for SoDPI framework and conventional approach based

TABLE 5. Metadata fields for Layer-7 protocol based traffic statistics service.

Fields	Description
`struct L7_Proto_Parsed_Meta{`	
`struct Base_Metadata base_parent_ids;`	Set of IDs against registered Service as per struct Base_Metadata
`uint32 client_ip, server_ip;`	Client and Server IP of stream
`uint16 source_port, dest_port;`	Source and Destination port
`eProtocols proto;`	L-7 application protocol
`struct timeval event_time;`	Time of respective L-7 application event
`struct L7_proto_fields {`	L-7 protocol specific field values in TLV form
`uint8 field_type;`	L-7 protocol extracted field type
`uint16 len;`	Length of field value
`uchar * value;`	L-7 field value
`L7_proto_fields *next;`	Next protocol field details
`}objL7_proto_fields;`	
`uint32 len_of_contents;`	Length of contents extracted for L-7 protocol events
`char * contents;};`	L-7 protocol data contents

DPIA systems, required data processing resources can be grouped into two categories (i) General set of resources required for base data processing system setup (ii) Custom set of resources required for DPIA packet processing. General set of resources include hardware components consisting of processing server chassis with CPU, memory and storage, arrangements for space, cooling, power and other auxiliary items along with licenses for operating system (OS) as software components. Custom data processing resources for DPIA processing consist of hardware components for high speed data provisioning along with high speed NICs for data access while software resources include licenses for DPIA related packet processing components.

For conventional approach, total processing resources $R^N_{Conv\_Total}$ that are required for DPIA processing functions at N number of standalone T-NMS systems for a given input data rate, can be expressed in terms of respective unitary component costs as follows:

$$R^N_{Conv\_Total} = \sum_{i=1}^{N} (R^i_{bs} + R^i_{prv} + R^i_{pc} + \varepsilon^i_{fc} R^i_{fc} + \varepsilon^i_{sr} R^i_{sr} + \varepsilon^i_{L7sa} R^i_{L7sa}) \quad (1)$$

where R_{bs} is unified unitary cost for general base system setup components, R_{prv} as unitary cost of resources required for

high speed data provisioning setup while R_{pc}, R_{fc}, R_{sr} and R_{L7sa} are unitary costs of DPIA components for packet capturing, protocol classification, stream reassembly and Layer-7 parsing and semantic analysis respectively. ε_{fc}, ε_{sr} and ε_{L7sa} are boolean value based requirement control variables for protocol classification, stream reassembly and Layer-7 semantic analysis DPIA services respectively. For conventional approach as shown in equation (1) required DPIA processing resources increase linearly with increase in the number of DPIA basedT-NMS systems.

For proposed SoDPI framework based approach, total required resources $R^N_{SoDPI\_Total}$ for same N client systems can be expressed as follows:

$$R^N_{SoDPI\_Total} = R_{bs} + R_{prv} + R_{pc} + \left(\varepsilon_{fc} \left|\varepsilon_{sr}\right| \varepsilon_{L7sa}\right) R_{fc} + \left(\varepsilon_{sr} \left|\varepsilon_{L7sa}\right.\right) R_{sr} + \varepsilon_{L7sa} R_{L7sa} \quad (2)$$

For equation (2), R_{bs}, R_{pc} and R_{prv} are mandatory components for a SoDPI framework based DPIA data processing setup. Similarly DPIA processing for L-7 semantic analysis requires protocol classified and stream reassembled data that shows R_{L7sa} is always accompanied with R_{sr} and R_{fc}, while R_{sr} is always accompanied with R_{fc}. To estimate maximum resources required for scenario where we need all DPIA services, we set values as true for ε_{fc}, ε_{sr} and ε_{L7sa}. With this assumption, $R^N_{SoDPI\_Total}$ for a given input data rate limit, can be expressed as follows:

$$R^N_{SoDPI\_Total} = R_{bs} + R_{prv} + R_{pc} + R_{fc} + R_{sr} + R_{L7sa} \quad (3)$$

As all components of SoDPI framework perform single time data processing for N number of T-NMS clients and deploys single instance of DPIA processing components, while for conventional standalone approach, N independent instances of same components are required that shows significant reduction in CAPEX with SoDPI framework. This is further evaluated and verified in section 7 showing comparison for both approaches.

VI. IMPLEMENTATION DETAILS OF SoDPI BASED EXAMPLE CLIENT APPLICATIONS FOR FRAMEWORK EVALUATION

For evaluation of proposed SoDPI framework, two DPIA based example monitoring applications are developed that avail different DPIA services. Example applications include (i) Protocol-aware statistics collector application and (ii) Data transfer monitoring application for SMTP and POP3 protocols. Each example application incorporates *SoDPI API client stub* for sending DPIA processing requests to SoDPI framework. *SoDPI API client stub* transforms SoDPI APIs based management, configuration and service requests into binary messages and transmits to SoDPI framework for processing of respective DPIA requests. Implementation details for each example application along with control and data processing flows are described in coming sub-sections. For evaluation of SoDPI framework capabilities to simultaneously serve multiple client applications, both example applications avail required different DPIA services concurrently as per setup given in Fig. 9. Performance results for both SoDPI framework based example applications are analyzed and compared in section 7.

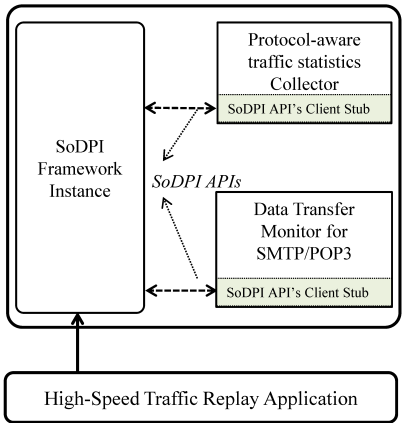

FIGURE 9. SoDPI framework setup with both example applications.

A. PROTOCOL-AWARE TRAFFIC STATISTICS COLLECTOR APPLICATION

Protocol-aware traffic statistics collector application avails *Layer-7 Protocol based traffic statistics* service of DPIA framework. It collects and displays protocol based flow statistics after required DPIA packet processing at SoDPI framework instance. It can be extended to develop multiple T-NMS applications including application-aware data usage calculation and charging systems for network service providers [12], [37] and application-aware traffic load balancing systems [9], [37]. Source code listing for part of this example application that interacts with SoDPI framework is given in Code Listing 1 in Appendix. As per given code listing, client establishes DPIA session with SoDPI instance at specified server address and listening port. After required configuration of client side storage location for results, client registers for classification statistics via *register_aggregated_stats_access_for_Protocols* API function call for all supported protocols specified as *ALL_PROTOS* with specified time interval for periodic transmission of statistics results. Service access flow depicting sequence of SoDPI API calls along with respective responses from SoDPI framework is shown in Fig. 10.

B. DATA TRANSFER MONITORING APPLICATION FOR SMTP AND POP3

Data transfer monitoring application is the second example application developed for evaluation of proposed framework that avails *Service for L-7 protocols extracted information access*for fields and contents of SMTP and POP3 protocols. After receiving DPIA processing results from SoDPI framework, it displays details for data messages and files exchanged via SMTP and POP3 protocols after required DPIA packet processing at SoDPI framework instance. This example application can be extended to develop Data leakage

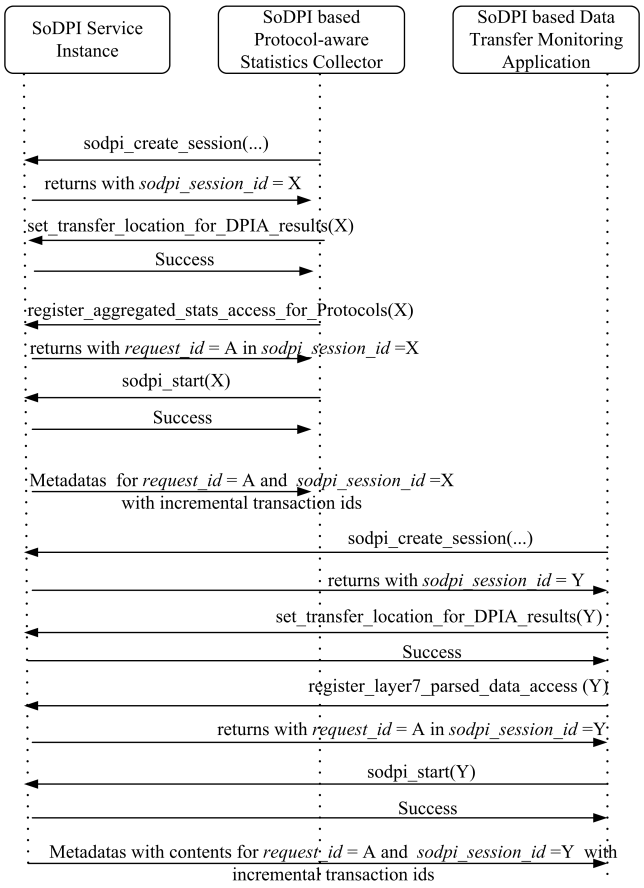

FIGURE 10. Services access flow with API function calls for both SoDPI based example applications.

Prevention Systems (DLPS) [6], Copyright Protection Systems [7] and Lawful interception (LI) systems [8]. Service access flow for data transfer application is given in Fig. 10 along with flow for first example application for application-aware flow statistics collector example application.

Source code listing for part of data transfer monitoring application that interacts with SoDPI framework is given in Code Listing 2 in Appendix. As per given code listing, after session setup and configuration , Layer-7 parsed data is requested via*register_layer7_parsed_data_access* API function call for SMTP and POP3 protocols with option of *MetadataWithContentsAccess.*

VII. EXPERIMENTAL EVALUATION

For evaluation and performance comparison of SoDPI framework with example monitoring applications mentioned in section 6, we have also implemented conventional approach based versions for both example monitoring applications. Experimental evaluations are performed for example application setups for both approaches with varying input data rates. System experimental evaluations firstly include comparisons of accuracy performance and consumed data processing resources in terms of peak CPU utilizations and peak memory utilizations. Later, detailed comparison is performed for required logistical resources including software, hardware and associated components for both approaches. Details for designs of conventional approach based example monitoring applications, evaluation environment, packet traffic trace, packets replay and comparison of results are given in following sub-sections.

A. IMPLEMENTATION DETAILS FOR CONVENTIONAL APPROACH BASED EXAMPLE MONITORING APPLICATIONS

Conventional approach based version of both example applications discussed in section 6, are developed with incorporation of independent DPIA processing components with respective application.

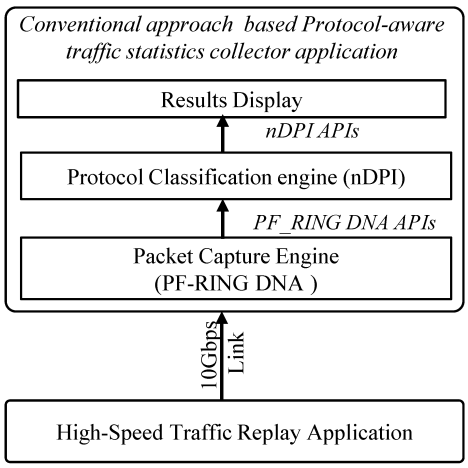

FIGURE 11. Conventional approach based design of protocol-aware statistics collector application with evaluation setup.

Setup for version of application-aware statistics collector example application implemented based on conventional approach is shown Fig. 11. This implementation incorporates dedicated DPIA related processing components including PF_RING DNA [15] for high-speed packet capturing and nDPI [22] for packet classification, in contrast to SoDPI framework based implementation where DPIA processing components incorporated in SoDPI framework, are shared by all client applications.

Version of data transfer monitoring example application based on conventional approach with setup detail is shown in Fig. 12. Implementation consists of dedicated integrated DPIA processing components including PF_RING DNA [15] for high-speed packet capturing, nDPI [22] for packet flow classification, Libnids [28] for stream re-assembly and custom SMTP/POP3 protocol analyzers for Layer-7 semantic analysis.

B. EVALUATION ENVIRONMENT

Experimental setup consists of two server machines. Server 1 is based on Intel S5500HCV chassis with Intel® Xeon(R) E5620 2.40 GHz 8-core dual processors, while Server 2 is based on Intel S5520HC chassis with Intel Xeon E52630

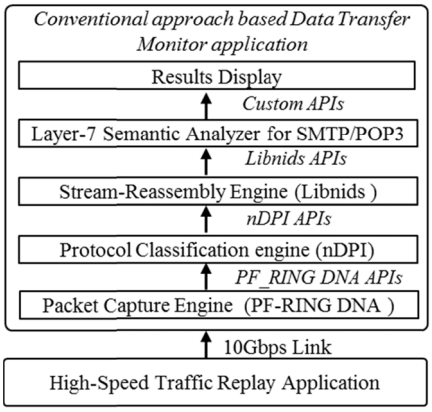

FIGURE 12. Conventional approach based design of data transfer monitoring application with evaluation setup.

2.27 GHz 8-core dual processors. Both servers are based on Ubuntu 14.04 LTS 64-bit OS while each server equipped with 16 GB DDR3 1333MHz, 1 TB SATA storage and Intel Dual-10GE X520-SR NIC adapter.

First server machine is used for performance evaluation of proposed SoDPI framework implementation with respective example applications and conventional standalone approach based example applications. Second server machine is used as traffic replay machine to play data packets for system evaluations.

C. PACKET TRAFFIC TRACE

Packet traffic trace is collected from backbone link of campus network and used for system evaluation. Aggregated data size of packet files is ∼5.0 GB with details of packets and flows mentioned in Table 6. Distribution of packet counts with respect to packet size is depicted in Fig. 13.

TABLE 6. PACKET traffic trace specifications.

Attribute	Value
Total Packet count	9,570,371
TCP Flows count	141,121
UDP Flows count	31,038
Average Packet Size *(Bytes)*	524

D. PACKETS REPLAY AND RECEPTION

To replay packets at multiple data rates for evaluation of SoDPI framework and conventional approach based example applications, we have used *pfsend* [36] utility. Transmission of packet trace files are performed for by varying data input rate from 2Gb/s to 8Gb/s for setups of SoDPI framework and conventional approach based example applications. Reception of packet data capturing is observed without any packet drops for multiple input data rates as PF_RING DNA [15] is used by all test setups as high speed packet capturing implementation.

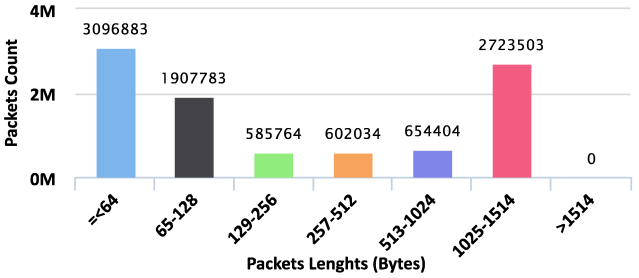

FIGURE 13. Packet-Length based distribution of packets.

E. EVALUATION DETAILS WITH COMPARISON OF RESULTS

We first evaluate standalone conventional approach based example applications with integrated DPIA stacks. Later we evaluate SoDPI framework based example applications in similar evaluation environment. Comparisons of evaluation results include accuracy performance, consumed data processing resources along with overall logistical resources required for respective implementations.

1) ACCURACY PERFORMANCE

Results for both versions of application-aware statistics collector application are given in Fig. 14 as distribution of packet flow counts for respective Layer-7 protocols. Results show that HTTP and SSL protocols are major part of packet traffic samples with 30,705 packet flows for HTTP protocol, 28,809 packet flows for SSL protocol without available SSL certificate information while 19,820 packet flows for SSL protocol traffic with SSL certificate information available. Protocols with less than 250 flows are collectively displayed as 'Others' category with 1,486 packet flows.

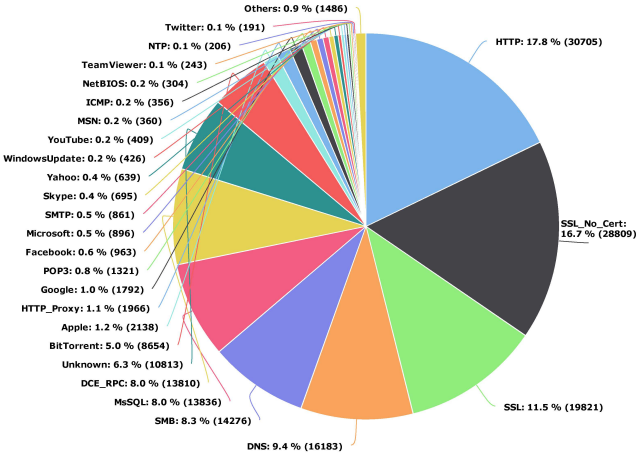

FIGURE 14. Protocol-wise distribution of packet flow counts.

Results for both versions of data transfer monitor applications for SMTP and POP3 protocols are shown in Fig. 15 with respective scaling down factors and measuring units for displayed parameters.

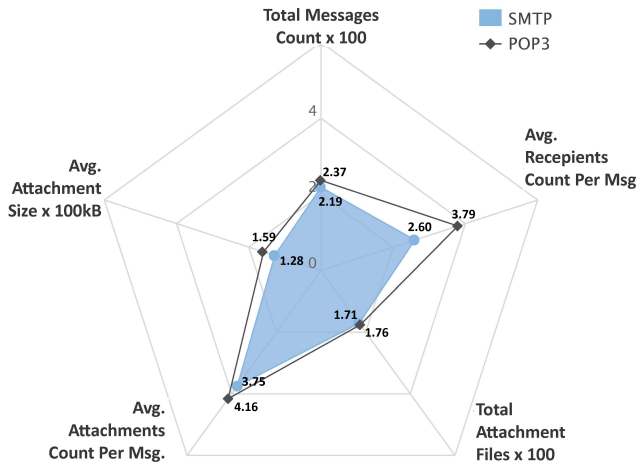

FIGURE 15. Collected statistics for POP3/SMTP based data transfer monitoring applications.

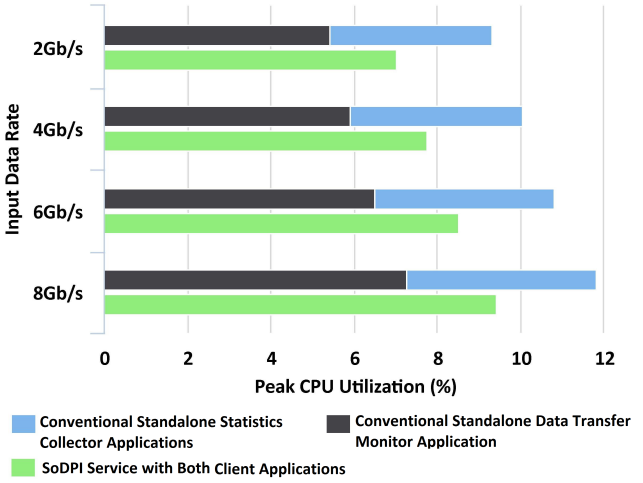

FIGURE 16. Comparison of CPU Utilizations.

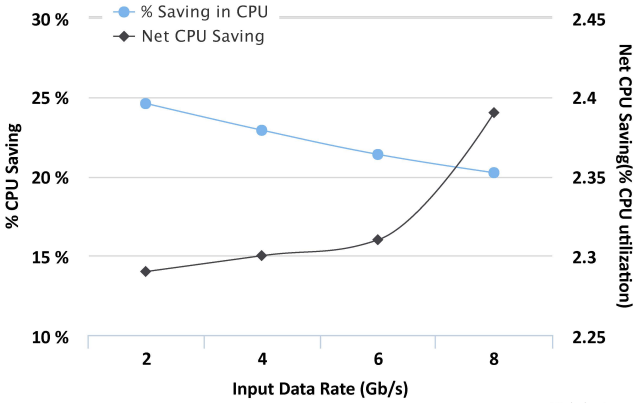

FIGURE 17. CPU Utilization savings with SoDPI framework.

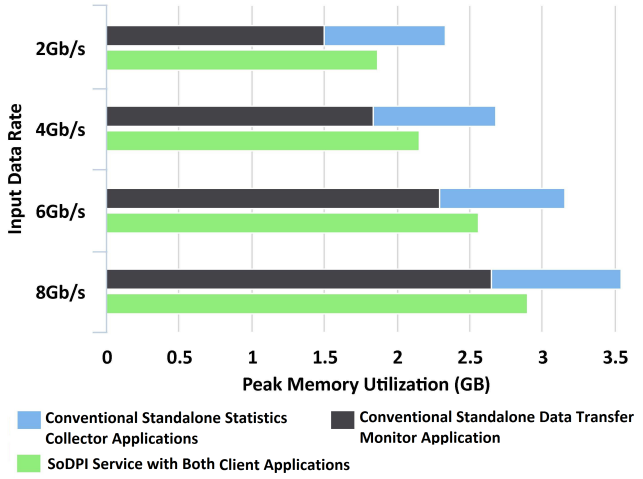

FIGURE 18. Comparison of memory utilizations.

L-7 protocol extracted information statistics show total messages counts, average recipients count per message, total attachments files exchanged, average attachment size in kilo Bytes (kB) and average attachment counts per message for both SMTP and POP3 protocols. Displayed results show 2,370 messages and 172 data file transfers extracted for POP3 protocols with average attachment size of 159kB, while there are 2,190 messages and 176 total attached data file transfers extracted for SMTP protocols with average attachment size of 128kB.

Accuracy performance evaluation produced same results for conventional approach based example applications and SoDPI framework based example applications that shows proposed framework did not affect accuracy performance for SoDPI based example applications.

2) COMPARISON OF CPU UTILIZATION AND MEMORY FOOTPRINTS

Another performance measurement of SoDPI framework is based on comparison of CPU and memory utilizations along with respective savings. For different input data rates, CPU utilizations for respective evaluation setups are shown in Fig. 16, while Fig. 17 highlights CPU savings with SoDPI framework. Comparison of system memory utilizations for respective evaluation setups is shown in Fig. 18 while memory savings with SoDPI framework is summarized in Fig. 19 for multiple input data rates.

Firstly, we compare both conventional standalone approach based applications to identify difference of CPU utilization and consumed system memory. As per Fig. 16 and Fig. 18, resource comparison for both conventional approach based example applications shows that peak memory and CPU consumptions by conventional approach based *data transfer monitoring application* are 2.65 GB and 7.25 % respectively at input data rate of 8 Gb/s, while conventional approach based *protocol-aware traffic statistics collector application* consumed only 0.898 GB memory and 4.57% CPU for same input data rate. *Data transfer monitoring application* consumes higher resources due to computationally expensive processing for stream data reassembly and Layer-7 semantic analysis of TCP flows for SMTP and POP3 protocols.

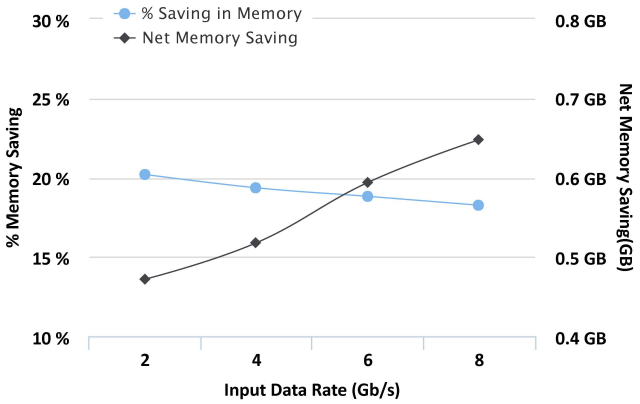

FIGURE 19. Memory utilization savings with SoDPI framework.

Secondly we compare SoDPI framework when simultaneously serving both SoDPI based example monitoring applications, with version of similar example applications based on conventional standalone approach. Results comparison shows that SoDPI framework consumed considerably less memory and CPU resources. With input data rate of 8Gb/s, SoDPI framework consumed 0.648 GB less memory and 2.39 % less CPU utilization than aggregated respective resources consumed by both conventional standalone approach based example applications. We also analyze savings with SoDPI framework in terms of percentage, with figure 17 depicting savings for CPU Utilizations and Fig. 19 showing savings for memory utilizations. Results show 18.3% saving in memory and 20.2% saving in CPU utilization with SoDPI framework for input data input rate of 8Gb/s. SoDPI framework utilized lower amount of CPU and memory resources because of single time DPIA processing for both SoDPI based example applications.

3) OVERALL LOGISTICAL RESOURCE REQUIREMENTS

Comparison of logistical resources is summarized in Table 7. Results show that SoDPI framework based approach requires single instance of each data processing component in comparison to conventional standalone approach that requires independent DPIA processing components, hardware/software components and data provisioning links for each example application.

As per Table 7, for conventional approach based standalone example applications, equation (1) specified in section 5 for two example applications with $N = 2$ becomes as follows:

$$R^2_{Conv\_Total} = 2R_{bs} + 2R_{prv} + 2R_{pc} + 2R_{fc} + R_{sr} + R_{L7sa} \quad (4)$$

As R_{sr} and R_{L7sa} are considerably small in comparison to sum of rest of four elements in (4), hence with considerably fair assumption, $R^2_{Conv\_Total}$ can be approximated as follows:

$$R^2_{Conv\_Total} \cong 2(R_{bs} + R_{prv} + R_{pc} + R_{fc} + R_{sr} + R_{L7sa}) \quad (5)$$

TABLE 7. Comparison of logistical resource utilization.

Data Processing Components	Conventional approach	SoDPI framework
General H/w and S/w Components *(Number of Units)*		
Base system Hardware along with OS (R_{bs})	2	1
Data Provisioning Links (R_{prv})	2	1
Software licenses for DPI components *(Number of Units)*		
Packet Capturing (R_{pc})	2	1
Packet Classification (R_{fc})	2	1
Stream reassembly (R_{sr})	1	1
Layer-7 Semantic Analysis (R_{L7sa})	1	1

While for SoDPI framework based example applications, $R^N_{SoDPI\_Total}$ in equation (2) specified in section 5 becomes as follows for $N = 2$:

$$R^2_{SoDPI\_Total} = R_{bs} + R_{prv} + R_{pc} + R_{fc} + R_{sr} + R_{L7sa} \quad (6)$$

Comparison of equation (5) and equation (6) based on Table 3 shows that:

$$R^2_{SoDPI\_Total} \cong \frac{1}{2} R^2_{Conv\_Total} \quad for\ N = 2$$

Above comparison of logistical resources for both approaches verifies that SoDPI framework requires reduced number of DPIA processing resources along with respective hardware and software components included in base system setup. This directly results in significant reduction of CAPEX in comparison to conventional approach for DPIA based T-NMS systems. Reduced number of hardware and software components also results in less maintenance and management that reduces OPEX as well for service providers.

VIII. CONCLUSION

This research work identifies duplicate packet processing of high speed data for required DPIA operations by multiple independent T-NMS systems once they are deployed in same enterprise network. This results in higher CAPEX and OPEX for network operators due to deployment of resources for duplicate DPIA processing. In this research work, a novel approach is described as 'Heterogeneous Service-oriented Deep Packet Inspection and Analysis (SoDPI)' framework along with its architecture, implementation and evaluation details. Proposed SoDPI framework exploits shared deployment of computationally expensive DPIA processing components, incorporates abstractions for diversified DPIA processing via SoDPI APIs and simultaneously provides heterogeneous DPIA services to multiple T-NMS systems. The SoDPI framework is based on multi-layer architecture consisting of three layers i.e. Core DPIA Processing (CDP) layer, DPIA Services Abstraction (DSA) layer and SMD layer for Service Management and Data Dispatching functions. SoDPI framework provides four different DPIA services that

include General Packet Traffic Statistics, Layer-7 protocols based traffic statistics, Reassembled data access of TCP streams and Access of extracted protocol fields and contents after semantic analysis of L-7 protocols. System evaluations include comparison of SoDPI framework with conventional standalone approach based monitoring applications for accuracy performance, computational resource utilization and required logistical resources. Evaluation results show that accuracy performance of SoDPI framework is at par with conventional approach based example applications but with reduced amount of required computational and logistical resources for DPIA operations. Comparing CPU and memory utilizations, SoDPI framework instance with two example monitoring applications consumed significantly reduced amount of computational resources in terms of peak CPU and memory utilization with 20.2% saving in CPU utilization and 18.3 % saving in memory utilization. Comparison of required logistical resources also show that independent instances of software and hardware components for DPIA processing are required by conventional approach based example applications while proposed SoDPI framework deploys single instance of hardware and software components for DPIA processing along with single data provisioning arrangement for multiple client applications. Reduced number of required hardware, software and associated components results in significant reduction of CAPEX as well as OPEX for service providers.

APPENDIX

```
int32  sodpi_sessid = sodpi_create_session
                      (SERVER_IP, SERVER_PORT);
int32 Resp = set_transfer_location_for_DPIA_results
                      (sodpi_sessid,CLIENT_IP,
                       META_STORAGE_LOCATION1,,.....);
int32 Req_Id =
register_aggregated_stats_access_for_Protocols(
                       sodpi_sessid,"0.0.0.0",
                       "255.255.255.255", 1,
                       ALL_PROTOS);
Resp = sodpi_start (sodpi_sessid);
```

Code Listing 1. Code listing for protocol-aware traffic statistics collector application.

```
int32  sodpi_sessid = sodpi_create_session
                      (SERVER_IP, SERVER_PORT);
int32 Resp = set_transfer_location_for_DPIA_results
                      (sodpi_sessid, CLIENT_IP,
                       META_STORAGE_LOCATION2,.....);
list_of_protocols * protos = new struct
                                list_of_protocols();
protos->protocol = eSMTP;
protos->next = new struct list_of_protocols();
protos->next->protocol = ePOP3;
protos->next->next = NULL;
Int32 RequestId = register_layer7_parsed_data_access
                      (sodpi_sessid,"0.0.0.0",
                       "255.255.255.255", protos,
                       MetadataWithContentsAccess);
Resp = sodpi_start (sodpi_sessid);
```

Code Listing 2. Code listing for Data transfer monitoring application.

TABLE 8. Definitions for common data structures and enumerations for SoDPI framework.

Fields	Description
`enum Direction { client_to_server, server_to_client, both_directions};`	TCP stream data direction
`enum ePortType { TCP,UDP};`	Transport Port Types
`enum eLevelOfAccess { MetadataAccessOnly, MetadataWithContentsAccess};`	Level of Access requested
`enum eProtocols { eSMTP, ePOP3};`	L-7 Protocol Types
`typedef struct { uint16 port; ePortType port_type; list_of_ports *next; } list_of_ports;`	List of TCP/UDP ports
`typedef struct { eProtocols protocol; list_of_protocols *next; } list_of_protocols;`	List of Protocols

REFERENCES

[1] J. L. Garcia-Dorado, A. Finamore, M. Mellia, M. Meo, and M. Munafo, "Characterization of ISP traffic: Trends, user habits, and access technology impact," *IEEE Trans. Netw. Service Manag.*, vol. 9, no. 2, pp. 142–155, Jun. 2012.

[2] C. Xu, S. Chen, J. Su, S. M. Yiu, and L. C. K. Hui, "A Survey on regular expression matching for deep packet inspection: Applications, algorithms and hardware platforms," *IEEE Commun. Surveys Tut.* vol. PP, no. 99, 2016.

[3] P. Yasrebi, S. Monfared, H. Bannazadeh, and A. Leon-Garcia, "Security function virtualization in software defined infrastructure," in *Proc. IFIP/IEEE Int. Symp. Integr. Netw. Manage. (IM)*, May 2015, pp. 778–781.

[4] Y.-D. Lin, P.-C. Lin, V. K. Prasanna, H. J. Chao, and J. W. Lockwood "Guest editorial deep packet inspection: Algorithms, hardware, and applications," *IEEE J. Sel. Areas Commun.*, vol. 32, no. 10, pp. 1781–1783, Oct. 2014.

[5] V. Jyothi, S. K. Addepalli, and R. Karri, "Deep packet field extraction engine (DPFEE): A pre-processor for network intrusion detection and denial-of-service detection systems," in *Proc. IEEE ICCD*, Oct. 2015, pp. 287–293.

[6] R. Tahboub and Y. Saleh, "Data leakage/loss prevention systems (DLP)," *Int. J. Inf. Sys.*, vol. 1, no. 1, pp. 13-19, 2014.

[7] M. Mueller, A. Kuehn, and S. Santoso, "Policing the network: Using DPIA for copyright enforcement," *Surveill. Soc.*, vol. 9, no. 4, p. 348–364, 2012.

[8] *ETSI-TR-102-528, Lawful Interception (LI) Interception domain Architecture for IP networks*. accessed Aug. 5, 2016. [Online]. Available: www.etsi.org

[9] C. S. Yang, "A network management system based on DPIA," in *Proc. 13th Int. Conf. Netw.-Based Inf. Syst. (NBiS)*, 2010, pp. 385–388.

[10] M. Jarschel, F. Wamser, T. Hohn, T. Zinner, and P. Tran-Gia, "SDN-based application-aware networking on the example of Youtube video streaming," in *Proc. 2nd Eur. Workshop Softw. Defined Netw.*, 2013, pp. 87–92.

[11] R. Serral-Gracià, E. Cerqueira, M. Curado, M. Yannuzzi, E. Monteiro, and X. Masip-Bruin, "An overview of quality of experience measurement challenges for video applications in IP networks," in *Proc. Wired/Wireless Int. Commun.*, 2010, pp. 252–263.

[12] F. Faria and J. M. Nogueira, "Context-based application-aware pricing for composite mobile services in wireless networks,"*Proc. Int. Conf. Intelligence in Next Generation Networks (ICIN)*, Berlin, Germany, Oct. 2010, pp. 1–6.

[13] L. Rizzo, L. Deri, and A. Cardigliano. (2012). *10 Gbit/s Line Rate Packet Processing Using Commodity Hardware: Survey and New Proposals*. [Online]. Available: http://luca.ntop.org/10g.pdf

[14] J. L. Garcia-Dorado, F. Mata, J. Ramos, P. M. S. D. Río, V. Moreno, and J. Aracil , *High-Performance Network Traffic Processing Systems Using Commodity Hardware*. Heidelberg, Germany: Springer, 2013, pp. 3–27.

[15] *PF_RING DNA: High-Speed Packet Capture, Filtering and Analysis*. accessed Aug. 5, 2016. [Online]. Available: http://www.ntop.org/products/packet-capture/p_ring/

[16] *PF_RING ZC: High-Speed Packet Capture, Filtering and Analysis With Zero Copy*. accessed Aug. 5, 2016. [Online]. Available: http://www.ntop.org/products/packet-capture/pf_ring/pf_ring-zc-zero-copy/

[17] L. Rizzo, "Netmap: A novel framework for fast packet I/O," in *Proc. USENIX Annu. Tech. Conf.*, 2012, p. 9.

[18] N. Bonelli, S. Giordano, G. Procissi, and L. Abeni, "A purely functional approach to packet processing," in *Proc. 10th ACM/IEEE Symp. Archit. Netw. Commun. Syst. (ANCS)*, New York, NY, USA, 2014, pp. 219–230.

[19] *Intel DPDK: Data Plane Development Kit*. accessed Aug. 5, 2016. [Online]. Available: http://dpdk.org

[20] A. Dainotti, A. Pescape, and K. C. Claffy, "Issues and future directions in traffic classification," *IEEE Netw.*, vol. 26, no. 1, pp. 35–40, Jan./Feb. 2012.

[21] *PACE: Protocol & Application Classification Engine*. accessed Aug. 5, 2016. [Online]. Available: https://ipoque.com/products/pace

[22] *nDPI: Open and Extensible LGPLv3 Deep Packet Inspection Library*. [Online]. Available: http://www.ntop.org/products/deep-packet-inspection/ndpi/

[23] S. Alcock and R. Nelson, "Libprotoident: Traffic classification using lightweight packet inspection," Univ. Waikato, Hamilton, New Zealand, Tech. Rep., 2012. [Online]. Available: http://research.wand.net.nz/software/libprotoident.php

[24] *L7filter: Application Layer Packet Classifier for Linux*. accessed Aug. 5, 2016. [Online]. Available: http://l7-filter.sourceforge.net/

[25] T. Bujlow, V. Carela-Español, and P. Barlet-Ros, "Cover image independent comparison of popular DPI tools for traffic classification," *Comput. Netw.*, vol. 76, pp. 75–89, Jan. 2015.

[26] A. Papadogiannakis, M. Polychronakis, and E. P. Markatos, "Stream-oriented network traffic capture and analysis for high-speed networks," *IEEE J. Sel. Areas Commun.*, vol. 32, no. 10, p. 1849, Oct. 2014.

[27] Snort: *Lightweight Intrusion Detection for Networks*. accessed Aug. 5, 2016. [Online]. Available: https://www.snort.org/

[28] Libnids: *Network Intrusion Detection System with IP Defragmentation, TCP Stream Assembly and TCP Port Scan Detection*. accessed Aug. 5, 2016. [Online]. Available: http://libnids.sourceforge.net/

[29] A. X. Liu, C. R. Meiners, E. Norige, and E. Torng, "High-speed application protocol parsing and extraction for deep flow inspection," *IEEE J. Sel. Areas Commun.*, vol. 32, no. 10, p. 1864–1880, Oct. 2014.

[30] Z. Li *et al.*, "NetShield: Massive semantics-based vulnerability signature matching for high-speed networks," in *Proc. SIGCOMM*, 2010, pp. 279–290.

[31] *Service-Oriented Architecture Standards*. accessed Aug. 5, 2016. [Online]. Available: http://www.opengroup.org/standards/soa

[32] V. Moreno *et al.*, "Multi- granular, multipurpose and multi-Gb/s monitoring on off-the- shelf systems," *Int. J. Netw. Manage.*, vol. 24, no. 4, pp. 221–234, 2014.

[33] A. Bremler-Barr, Y. Harchol, D. Hay, and Y. Koral, "Deep packet inspection as a service," in *Proc. 10th ACM Int. Conf. Emerging Netw. Experiments Technol.*, 2014, pp. 271–282.

[34] SMTP: *Simple Mail Transfer Protocol*. accessed Aug. 5, 2016. [Online]. Available: https://www.ietf.org/rfc/rfc2821.txt

[35] *POP3: Post-Office Protocol*. accessed Aug. 5, 2016. [Online]. Available: https://www.ietf.org/rfc/rfc1939.txt

[36] *PFSEND: Wirespeed Traffic Generation*. accessed Aug. 5, 2016. [Online]. Available: http://www.ntop.org/solutions/wire-speed-traffic-generation/

[37] ITU-T Rec. Y.dpifr. *Framework for Deep Packet Inspection*. accessed Aug. 5, 2016. [Online]. Available: https://www.itu.int/rec/T-REC-Y.2771/

MUHAMMAD ASRAR ASHRAF received the bachelor's degree in electronics engineering from the NED University of Engineering and Technology, Karachi, in 1999, and the M.S. degree in computer engineering from the University of Engineering and Technology, Taxila, Pakistan, in 2005, where he is currently pursuing the Ph.D. degree. He has over 16 years of industrial experience and has worked for Communications Enabling Technologies Inc. and the Center for Advanced Research in Engineering, Pakistan. His areas of expertise include design and development of VoIP systems, embedded systems, real-time system applications, and packet processing systems for high-speed networks.

HABIBULLAH JAMAL received the B.Sc. degree from the University of Engineering and Technology, Lahore, Pakistan, in 1974, and the M.A.Sc. and Ph D. degrees from the University of Toronto, Canada, in 1979 and 1982, respectively, all in electrical engineering. He has served academia throughout his professional career. His longest association was with the Department of Electrical Engineering, University of Engineering and Technology Taxila, Pakistan, from where he retired as a Tenured Professor in 2012. He remained the Dean, FE&EE, from 1994 to 2000 and the Vice Chancellor of UET, Taxila, from 2001 to 2009. He has been a Professor with the Faculty of Engineering Sciences, Ghulam Ishaq Khan Institute of Engineering Sciences Topi, KPK Pakistan, since 2012. He is a recipient of prestigious awards; eighth TERADATA National IT Excellence Awards for Excellence in IT Education in2008, the Performance Excellence in Engineering Award from the Institution of Engineers Pakistan on the Engineers Day in 2007, the ninth Pakistan Education Forum, the National Education Award in 2003, and the National Book Council of Pakistan Award in 1991. He has authored overtwo text books and 128 research papers. His research interests include digital design, analog and digital signal processing, and communications. He is a Fellow of many professional bodies.

SHOAB AHMED KHAN received the Ph.D. degree in electrical and computer engineering from the Georgia Institute of Technology, Atlanta, GA, USA, in 1995. He is currently a Professor and the Head of the Department of Computer Engineering with the College of Electrical and Mechanical Engineering, National University of Sciences and Technology, Pakistan. He has been actively involved in research and development. He has authored over 200 publications in international conferences and journals, and holds six U.S. patents. He has over 20 years of industrial experience in companies such as Scientific Atlanta, Picture Tel, Cisco Systems, and Avaz Networks. He designed a high density media processor for carrier class voice processing system as a Chief Architect. He is also the Founder of the Center for Advanced Studies in Engineering and the Center for Advanced Research in Engineering, two prominent organizations working for the promotion of research and development in Pakistan. He was a recipient of the National Education Award 2001 in the category of Outstanding Services to Science and Technology, the NCR National Excellence Award in the category of IT Education, the prestigious Cisco System Research Grant, and ICT Research and Development and PTCL Research and Development research funding.

ZAHEER AHMED received the B.E. degree in electronics engineering from NED University, Karachi, the M.S. degree in computer engineering from the National University of Sciences and Technology, Rawalpindi, the M.Sc. degree in nuclear power plant technology from KINPOE, and the Ph.D. degree in electronics engineering from the University of Engineering and Technology Taxila, Pakistan, in 2009. He has over 20 years of industrial experience in the area of hardware and software design and development. His areas of expertise are embedded systems, industrial control applications, ASIC design, cryptography, and software design and development.

MUHAMMAD IRAM BAIG received the M.S. and Ph.D. degrees in electrical engineering from UET, Taxila, Pakistan, in 1996 and 2010, respectively. He has served academia throughout his professional career. He is currently a Professor with the Department of Computer Engineering, University of Engineering and Technology, Taxila, Pakistan. His areas of expertise are digital design, testing/verification, and embedded system.