Deep Semantics Inspection over Big Network Data at Wire-Speed

Chengchen Hu, Xi'an Jiaotong University, Science and Technology on Information Transmission and Dissemination in Communication Networks Laboratory
Hao Li, Xi'an Jiaotong University
Yuming Jiang, Norwegian University of Science and Technology
Yu Cheng, Illinois Institute of Technology
Poul Heegaard, Norwegian University of Science and Technology

Abstract

Deep Semantics Inspection (DSI) proposed in this article reveals the semantics behind big network data on the fly. The key idea of DSI is to obtain a sketch for user behavior at wire speed, whose size is several orders of magnitude smaller than that of the raw data. Then, semantics analysis is applied to the obtained sketch. To demonstrate the use of DSI, this article also presents several practical user scenarios leveraging on the DSI system designed.

1. Introduction

Cisco Visual Networking Index report forecasts that the global Internet traffic will reach 1.0 ZB per year in 2015. Huge Internet traffic, produced by popular mobile applications, web services and social media, is a special type of big data that possesses the "4V-characteristics" (variety, velocity, volume and veracity) [1], termed as "big network data" in this article. If being properly managed, such data could reveal many opportunities and solve problems that have not been feasibly addressed or properly handled before.

Analysis on big network data is challenging since one has to handle massive and rapidly increasing amount of data from possibly many different sources. Current solutions for inspection of semantics is to record everything and then do post-processing and data mining offline. This is not a scalable solution. Lower level online (real time) traffic identification using Deep Packet Inspection (DPI) and Deep Flow Inspection (DFI) based on protocols, applications, *etc.*, is coarse-grained and inflexible, and will not give sufficient insight in the users' behavior and preferences. In this article we describe an approach to gain deep understanding of users' behaviors and preferences extracted from Protocol Data Unit (PDU) and the relationship between PDUs, and the agility to easily support various analysis purposes.

Specifically, we propose "Deep Semantics Inspection (DSI)", where the *semantics* is defined as the meanings and indications of a user's intent behind big network data. Relying on deep understanding of the internal relations between PDUs and their semantics, DSI extracts concise but yet descriptive meanings from the data at wire speed.

We use the example in Figure 1 to illustrate how DSI, DPI/DFI and offline data mining solutions work. There are two users: one uses laptop accessing Amazon and Facebook, and the other activates iPhone applications for Facebook and WeChat.

DSI discloses fine-grained information about the user, e.g., for the example in Figure 1, after using Chrome on an iPhone to view a SONY TV set on Amazon on Jan. 5, 2015, the user purchased the viewed TV set. In other words, leveraging on the descriptive information of "who, when, what, how, ...", DSI keeps the flexibility to further deduce more comprehensive analyses such as user profiling, discovering preference of TV set brands, mobile devices, browsers market share, *etc*. All these tasks can be completed at wire-speed. This enhances the output from DPI, which only conducts protocol or application level inspection, for example, HTTP flow, visiting Amazon/Facebook, and using WeChat. The content providers, Amazon, Facebook and WeChat in this example, would employ data mining on logs for various analysis, but normally in an offline manner. In addition, DSI performs on the raw Internet traffic, while the offline methods need to access the proprietary server logs, which is usually only practical for individual content providers or data holders such as Google or Amazon.

Figure 1. An illustrative example to compare DSI and DPI/DFI.

2. Inspecting Semantics over Big Network Data

In this section, we present in details the features and the design goals of DSI. To compare with related work, the design spaces are illustrated in Table 1.

Table 1. Design space of DSI.

DSI should be operating at (close to) wire speed. The wire speed in the core has been continuously increasing and meanwhile the volume of network data has dramatically increased. Combined with the need to correlate data from different network levels and sources, this poses great challenges to extract desired / useful values from low-value-density data. Data mining methods and tools are able to get deep understanding of static big data, such as off-line logs or semantic web, but they are usually not designed for streaming processing on big network data. Even when real-time analysis is not strictly needed for some cases, an offline analysis is always limited by storage for big network data, because the analysis capabilities are slower than the rate at which data is produced [6]. As a result, DSI explores a more appealing approach to only extract and store the information that is useful for further analysis [2].

DSI is designed to exhibit fine-grained semantics. Extracting and combining data from network, application and semantic levels, gives different insight and information about the state of the network, e.g., for management purposes, and understanding and knowledge about the users' behavior and preferences, e.g., for service customization purposes. DSI tries to report more logical intents about who, how, when, why, *etc*., besides the "what is a packet/flow" question answered by DPI/DFI. More and more applications integrate multiple functionalities or contain quite different contents with diverse data formats. For one example, people use Facebook to share pictures, update timeline, chat with friends, *etc.* For another example, users visit Amazon's web page, view product descriptions and other users' comments, compare prices, buy items, *etc.* DSI reports the specific (user) behaviors instead of protocols or applications, unlike what DPI/DFI does.

DSI should be agile to support various analysis targets based on the same input data. In other words, the output semantics of DSI should allow the flexibility of being redefined to obtain the interested information. This is similar to how we classify a flow by some unique features: A specific flow can for example be a series of packets with the same destination IP address, or packets with the same source-destination IP address pair. In fact, DSI outputs different granularity semantics in the example of Figure 1, from preliminary semantics like "**who** use(s) **what** visit **where** on **when**", to more complex semantics like "market share of browsers". On the contrary, DPI and DFI are usually fixedly designed with specific protocols and do not have the ability to flexibly change the identification purpose.

To the best of our knowledge, the DSI approach is the first study to extract semantics from big network data in a fine-grained, flexible and online manner.

3. Design and Implementation of a DSI System

We have designed and implemented a DSI system called Semantics On-Line Intent Detection (SOLID) as shown in Figure 2. For design details, please refer to [15].

Figure 2. The system architecture of SOLID. The specifications, system architecture and the data flow, are shown in the left-hand, the middle, and the right-hand side, respectively.

Data-flow of SOLID

Basically, SOLID deduces the semantics over three main stages. It first transforms the raw PDU into "application sketch" (app-sketch), which is a set of <field: value> pairs (e.g., **Time:** Jan. 5, 2015; **Host**: amazon.com; **Action**: view item, *etc.*). Next, SOLID works on the app-sketches to reveal the "behavior sketch" (behav-sketch). The behav-sketch is a set of minimized meaningful structured data describing user behavior (e.g., A user views a **Sony TV set** on **Amazon** at **time**, **day**, **month**, **year** using **iPhone**). Finally, we can infer the high-level semantics by applying user-flexible analysis over the large group of the user behaviors. A detailed data flow example is shown in the right-hand side of Figure 2. With the processing in SOLID, the data volume decreases tremendously step by step (PDU>app-sketch>behav-sketch>semantics). During this process, we have designed expressive specification, agile user space, and fast kernel space to achieve the three goals aforementioned, respectively.

Expressive Specification

We propose two specifications to express the applications and behaviors, respectively. First, app-spec is used to extract app-sketch, which is a pre-defined specification of the application protocols to parse a packet up to the application layer. It is generally in a form of Backus-Naur Form (BNF) with a set of production rules. The bottom left side of Figure 2 is an example of app-spec, which consists of several production rules within header and payload to illustrate an HTTP protocol. More specifically, the production starts from *S*, which produces the HTTP request *Q* and the response *E*. *Q* will further produce the request line *R* and a set of header fields *F*. Following the similar process, we can eventually get a whole HTTP protocol with the interested information, such as the catalog of the Amazon items. In addition, behav-spec is employed to extract interested properties from app-sketch, which is listed in the left-hand side of Figure 2 above the app-spec. The behav-spec is a set of key-value pairs indicating the deduced information. For example, we match the User-Agent field in HTTP-Amazon protocol to check whether this request was performed by the Chrome browser on an iPhone (User-Agent: iPhone.*Chrome).

The SOLID system relies on the app-spec and behav-spec for accurate results. It is easy to generate the specifications for public and well-defined described L7 protocols, but for applications using own proprietary protocols in the application layer, elaborate efforts are required to synthesize the specifications. The following principles are used to generate specifications in SOLID deployment.

• Standard public protocols often clearly define the meanings of the fields and values. For instance, in HTTP protocol, "Host" field is often used to differentiate web applications, and the behaviors can be inferred from the "URI" field. Other fields such as "User-Agent" and "Referrer" are also involved to describe the app-spec and behav-spec.

• "Proprietary protocols" are commonly defined with a user payload header in each application message, which are indications of user semantics. A proprietary protocol usually contains a protocol identifier (to distinguish it from other protocols), a user identifier (to indicate different users), and a behavior identifier (to denote the user action) with special separators (to separate different fields and the real data).

• For efficient handling process on the server side, applications tend to use structured format to carry their semantics, such as JSON/XML. These marked up languages are expressive and can be easily resolved.

Agile User Space

The architectural design of SOLID is illustrated in the middle column of Figure 2. We view the top layer as the user space, which issues the flexibility of the system and fills the gap between behavior sketch and semantics. The data volume is significantly reduced from the raw PDU to the behavior sketch in the kernel, and the unstructured big network data has been normalized into a unified format. In the user space, SOLID finally invokes the scenario-dependent analysis to conduct the interested semantics.

To be specific, SOLID abstracts a data set between the kernel and the user space as the behavior sketches (southbound interface), and meanwhile provides a set of unified APIs in the user space to query the sketches from comprehensive semantics calculations (northbound interface). Tools of data mining and information visualization can be integrated into the user space and the requirement is to comply with the unified interfaces. For example, by clustering a large group of user behaviors, we can obtain the correlations of different applications.

In general, with the minimized behavior sketches, we can conduct various analyses in the user space agilely. The user space of SOLID issues the flexibility for different applications to produce their own interested semantics based on the unique framework of SOLID, as well as the behavior sketch input. In the next section, we will present three practical scenarios to demonstrate the potentials of the user space.

Fast Kernel Space

The design of the kernel space determines the performance of SOLID to achieve the wire-speed processing goal.

The bottom layer in the SOLID's architecture named as Semantics Parsing Engine (SPE) resolves the reassembled PDUs into the application sketch according to the app-specs. SPE transforms the PDUs into the structured application sketch and reduces the data volume by ignoring the irrelevant payload. The SPE in SOLID first combines multiple app-specs into a distinguishable automaton, and a one-time parsing on this automaton can identify the protocol and extract the field values simultaneously to assure the high speed processing of the SPE. Please refer to our previous work [7] for the detailed design of the parsing method. Previous literatures are not sufficient for our purpose. For example, Binpac [8] extracts "http-request", "http-request-header", "http-response-body", but cannot go deep into the payload of the response. In addition, the flexible definitions result in overlaps between multiple app-specs, since they may be based on the same L7 protocols. Other related works identify and parse protocols separately [8, 9]. In particular, they either sequentially parse each app-spec, which is obviously not scalable, or set an inaccurate prior identifier to identify the protocol first, which risks the accuracy of the whole system [7].

Next, a Semantics Matching Engine (SME) is the middle layer of the SOLID design, which compares the application sketch with the predefined behav-spec and outputs the behavior-sketch. The DSI system is expected to scale with emerging specifications resulted from the growth of new applications/functions. We have proposed Rule Organized Optimal Matching (ROOM) in [10] to improve the matching performance-cost ratio by 1.5-23 times. The idea is to only activate a small subset of rules that could be possibly matched in each field, which avoids the intersection calculation of the candidate matched rules from each field, and increases the memory consumption by splitting one large matching structure into several much smaller ones. We have later extended ROOM to MP-ROOM [11] to support multiple PDUs for more complex behav-specs, which is used as the SME of SOLID. Intrusion detection system (IDS) [12] also has matching component to detect intrusions, which however cannot be directly leveraged in SOLID for two reasons. First, IDS is designed for security issues and is not flexible enough to work with the behav-spec. Though the number of intrusions is increasing, the growing speed of the matching rules is much slower because one vulnerability-based rule can express multiple instructions [12]. Second, only a few flows related to intrusions would be matched in an IDS, but every flow should have a match in SOLID, increasing the processing pressure of the SME.

4. Practical Cases

SOLID has been deployed in several practical scenarios and three of representative ones are presented below. These cases are respectively derived from on-line analysis over the big network data in a network service provider (NSP)'s network, a university's network, and a company's Intranet. There may be alternative ways to achieve the goal for each of the scenarios, but the merit shown here is the flexibility of SOLID: The unique framework and the same kernel space can be utilized by different applications to satisfy various analysis requirements.

Application Diagnosis

Figure 3 shows the deployment of SOLID in the Intranet environment for the diagnosis purpose. There is a web service provided by three equal servers (S_1-S_3), and N_1 is a load balancing node dispatching requests to the servers according to CPU utilizations in S_1-S_3. The SOLID system analyzes the mirrored network data in and out of N_1 and reports any exceptions. In this case, the diagnosis application in the user space is inputted with processing behavior between each server and users at wire speed and outputs the semantics whether one server works regularly or not according to the behavior transitions. The processing logic in the user space of SOLID continuously monitors the network traffic transactions' states between the servers and the users (behavior sketch), and based on this, it detects the abnormal transitions between states and locates the causes of (potential) problems.

It happened someday during the deployment of SOLID that S_2 failed to respond the correct data due to a disk error. S_2 cannot provide any service, but N_1 kept on dispatching new requests to S_2 due to its low CPU utilization. Traditional diagnosis tools did not activate any alarm in this case, because the network interface of S_2 was still up and the CPU/memory utilization was normal. In contrast, SOLID provided higher-level intelligence with app-spec and behav-spec, and reported in this case about the incomplete transaction of S_2... As indicated from the bottom half of Figure 3, SOLID monitored three behaviors in the servers. And normal nodes S_1 and S_3 experienced large success rate and small response time, while S_2 failed to respond from behavior 2 and did not trigger the request of behavior 3.

Figure 3. SOLID for the diagnosis purpose

Consulting Analysis

Traditionally, consulting companies use host-based methods, *e.g.*, embedding plugins, to collect data crossing different content providers (CPs), which, however, can be easily polluted by the unstable proportions of users/applications using such methods.

SOLID is capable to draw a macro picture of the operating situation in an NSP's network. Here we list one practical example, where SOLID filters out the access traffic of three video CPs in the *NSP's* network and generates the statistics in details. The output semantics in this case is the competition analysis based on the statistics of behavior sketches.

Figure 4(a) shows the overview of the traffic and user share of the three CPs, where multiple page-views from a single IP address only contribute one count. CP3 attracts the most users with the least traffic. The dominant user share infers its advantages of attracting users, but the low traffic raises a potential problem, as is how to make users stick to it. To understand the problem better, Figure 4(b) shows the statistics of CP3 in details. The "VOD" channel attracts most of the users, which however produces not so much traffic. In other words, users do not pay much time or money on watching the whole video but just had a glance at them. An implication is that there could be a risk of losing users if CP3 cannot provide interesting/attracting content. In addition, Figure 4(c) illustrates the users' clicking pattern in CP3's "VOD" channel. Each circle is a web page and its diameter is proportional with the page-views on this page. We could infer many suggestions from this graph, *e.g.*, which video attracts the most users? How many users actually pay for the video when they jump into the detailed descriptions? In this case, users fall away in WebPage2, which lowers the traffic in WebPage4. Usually, CP can perform such analysis with the web logs individually, but SOLID can do such mining at the network side for a much quicker response. More importantly, an NSP can compare the data from multiple CPs in its network, which is the advantage over the existing off-line analysis on the weblogs from a single CP.

Figure 4. The overview and detailed analysis of the three CPs with SOLID.

Correlation Analysis

The interrelationships between different applications are complex, since the developers may integrate diverse functionalities to enhance them. As a result, some of them overlap with each other: the QQ Messenger (the most popular IM client in China) promotes the news to its users and Twitter as an SNS (social network service) platform supports one-to-one chatting. On the other hand, some applications are complementary: a news-feed application can share the information with your friends on QQ or Twitter. With the correlation analysis performed by the network provider, the output semantics can help better understand the user preferences.

Figure 5. The application correlation analysis.

We demonstrate one-day frequencies of several popular applications in an NSP's network and evaluate their correlations in Figure 5, where X and Y axis are the normalized frequencies of using the corresponding applications. Figure 5(a) shows the positive correlation between QQ and Tencent Microblog. Since QQ has embedded some functions of Tencent Microblog, Tencent Microblog can gain its popularity from QQ. Figure 5(b) shows the negative correlation between Weibo and Tencent Microblog. From the figure, we have the following observations. 1) The two products are competitors in the microblog field. 2) They both have strong user loyalty, i.e., most users only use one of them and stay with it. We believe, such reports are valuable to the related companies and vendors for them to drive the right business decisions.

5. Performance Evaluation

Although SOLID outputs consistent and solid results as we demonstrated above, we aim to test the potential processing capacity and overhead. We evaluate SOLID's kernel space on an x86 platform with 12-core Xeon E5-2620 2 GHz and 32GB memory. We preload two real traces with their original orders of the segments. One was captured from the campus network of a university in China obtaining 4.5GB traffic. The other 7.5GB trace was collected at a Radio Network Controller (RNC) in Hangzhou, China. We implement 38 application specifications, including seven catalogs of the applications, such as SNS, media, online shopping, *etc.* We further give 1048 behavior specifications for the evaluations.

The single thread implementation of SOLID achieves 3.0Gbps and 2.7Gbps with the two real traces, respectively, i.e., about 1.7 times faster than NetShield [12]. It is reported in [12] that NetShield reaches the analysis speed of 11Gbps on a DARPA trace, while the throughput of SOLID on the same trace is 16.9Gbps. With multiple-thread evaluation using 10 cores, SOLID's throughput is 17.2Gbps for real trace one and 15.9Gbps for the second. Considering the throughput of NetShield is measured without reassembly work and only for HTTP protocol with less (794) rules, we believe that SOLID would achieve better performance than NetShield in a real network with parallel acceleration.

The memory costs scale with the number of cores. When 10 cores are used, the memory cost varies between 708MB and 839MB during the test with difference real traces. In the experiments, the compression ratio between the volume of raw big network data and the size of user sketch reaches 1216~1362. As a result, the data volume for further user-defined high level analysis can be significantly reduced to bridge the aforementioned enlarging rate gap over the big network data.

6. Discussions

This article introduces the concept of Deep Semantics Inspection (DSI), as well as its system framework to analyze big network data. As an initial work of DSI, there are several limitations and open questions that need to be explored in the future.

• The app-sketch and behav-sketch are extracted by app-spec and behav-spec, which are now manually generated according to the principle aforementioned. It is highly desired to study the (semi-) automation of this process, especially in the case that the application changes frequently. A previous study [13] investigated the automatic generation of the string/regex-based specifications, which yield insights for this problem.

• Current design in this paper is not capable to analyze encrypted traffic. Technically, using DSI over the raw, unencrypted traffic by decoding the SSL protocol is feasible through shadow agent nodes. Here, we remark that abuse of DSI allowing cross-referencing of individual's Internet activities is socially controversial. The inspection of each individual is usually forbidden due to privacy protection, but the knowledge of the macro activities should be helpful and government and Intranet censorship always exists legally. How to make balance between traffic analysis and privacy is an interesting topic and a recent work BlindBox inspecting the encrypted traffic is viewed as a start [14].

• The behav-sketch is the interface between the kernel and user space. A concise and efficient abstraction should be carefully designed. A simple illustration of vast ontology and imprecise concepts would be a nightmare. Starting from the study of a minimized but descriptive enough sketch categories, like who, when, where, subjects, objectives, actions, *etc.*, would be feasible.

7. Conclusion

In this article, we advocate the inspection of semantics over big network data. DSI/SOLID is designed to capture, analyze and present the semantics of the user intents by gathering the unstructured data into a unified framework. Three real cases leveraging on DSI/SOLID, as well as the performance experiments on high throughput and efficient memory usage, have demonstrated the usage and feasibility of DSI.

Acknowledgement

This paper is supported by the NSFC (61272459, 61221063), Program for New Century Excellent Talents in University (NCET-13-0450), ERCIM "Alain Bensoussan" fellowship program, Jiangsu Future Internet Innovation Project (BY2013095-1-12), open project of Science and Technology on Information Transmission and Dissemination in Communication Networks Laboratory (ITD-U14001/KX142600008). It is also supported by the NTNU QUAM (Quantitative modeling of dependability and performance) research lab (https://www.ntnu.edu/telematics/quam).

References

[1] "Big Data 101: Unstructured data analytics." [Online]. Available: http://www.intel.com/content/www/us/en/bigdata/unstructureddataanalyticspaper. html.

[2] C. Hu and Y. Jiang, "Online Semantic Analysis over Big Network Data," *ERCIM News*, no. 101, Apr. 2015.

[3] T. Benson, A. Anand, A. Akella, and M. Zhang, "MicroTE: Fine Grained Traffic Engineering for Data Centers," in *CoNEXT*, New York, NY, USA, 2011, pp. 8:1–8:12.

[4] "Market share for mobile, browsers, operating systems and search engines," 2013. [Online]. Available: http://www.netmarketshare.com/.

[5] "w3cSchools Browser Statistics and Trends," 2013. [Online]. Available: http://www.w3schools.com/browsers/browsers_stats.asp.

[6] "Cisco Visual Networking Index." [Online]. Available: http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white_paper_c11-520862.pdf.

[7] H. Li, C. Hu, J. Hong, X. Chen, and Y. Jiang, "Parsing Application Layer Protocol with Commodity Hardware for SDN," in *Proceedings of the Eleventh ACM/IEEE Symposium on Architectures for Networking and Communications Systems*, Oakland, CA, 2015, pp. 51–61.

[8] R. Pang, V. Paxson, R. Sommer, and L. Peterson, "binpac: A yacc for writing application protocol parsers," in *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, 2006, pp. 289–300.

[9] C. Meiners, E. Norige, A. X. Liu, and E. Torng, "Flowsifter: A counting automata approach to layer 7 field extraction for deep flow inspection," in *INFOCOM, 2012 Proceedings IEEE*, 2012, pp. 1746–1754.

[10] H. Li and C. Hu, "ROOM: Rule Organized Optimal Matching for fine-grained traffic identification," in *INFOCOM, 2013 Proceedings IEEE*, 2013, pp. 65–69.

[11] H. Li and C. Hu, "MP-ROOM: Optimal Matching on Multiple PDUs for Fine-Grained Traffic Identification," *Sel. Areas Commun. IEEE J. On*, vol. 32, no. 10, pp. 1881–1893, Oct. 2014.

[12] Z. Li, G. Xia, H. Gao, Y. Tang, Y. Chen, B. Liu, J. Jiang, and Y. Lv, "NetShield: Massive Semantics-based Vulnerability Signature Matching for High-speed Networks," in *Proceedings of*

the ACM SIGCOMM 2010 Conference, New York, NY, USA, 2010, pp. 279–290.
[13] Y. Wang, X. Yun, M. Z. Shafiq, L. Wang, A. X. Liu, Z. Zhang, D. Yao, Y. Zhang, and L. Guo, "A semantics aware approach to automated reverse engineering unknown protocols," in *20th IEEE International Conference on Network Protocols (ICNP)*, 2012.
[14] J. Sherry, C. Lan, R. A. Popa, and S. Ratnasamy, "BlindBox: Deep Packet Inspection over Encrypted Traffic," in *Proceedings of ACM SIGCOMM*, 2015.
[15] C. Hu, H. Li, Y. Jiang, P. Heegaard, Y. Cheng, J. Hong, G. Li, Q. Guo, S. Wu, Tech. Report: Deep Semantic Inspection for On-Line Analysis of Big Network Data, Tech. Report, Xi'an Jiaotong University, 2015. [Online]. Available: http://nskeylab.xjtu.edu.cn/people/huc/Pub/DSI_report.pdf .

Biographies

Chengchen Hu (M’09) received his Ph.D. degree from Tsinghua University, China, in 2008. He is currently an Associate Professor with the Department of Computer Science and Technology, Xi’an Jiaotong University, China. His research interests include network measurement, data center networking, software defined networking, etc. He is recipient of a fellowship from the European Research Consortium for Informatics and Mathematics (ERCIM), New Century Excellent Talents in University awarded by Ministry of Education, China.

Hao Li (S'13) received the B.S. degree in Computer Science from Xi'an Jiaotong University in 2010, and is now a Ph.D. candidate of the Department of Computer Science and Technology, Xi'an Jiaotong University. His research interests are network measurement and software defined networking.

Yuming Jiang (SM’14) received his bachelor's degree from Peking University, China and PhD degree from National University of Singapore (NUS). Since 2005, he has been a Professor with Norwegian University of Science and Technology (NTNU). His main research interest is in the provision and analysis of quality of service guarantees in communication networks. He was General Chair of IFIP Networking 2014 Conference and is author of the book ”Stochastic Network Calculus”.

Yu Cheng (SM’09) received his Ph.D. degree in electrical and computer engineering from the University of Waterloo, Canada, in 2003. He is now an associate professor in the ECE Department, Illinois Institute of Technology. His research interests include wireless networks, network security, and next-generation Internet technology. He received several Best Paper Awards including a Runner-Up Best Paper Award at ACM MobiHoc 2014. He received the NSF CAREER Award in 2011 and IIT Sigma Xi Research Award in the junior faculty division in 2013.

Poul Heegaard (SM´14) received his PhD in Telematics from the Norwegian University of Science and Technology (NTNU) in 1998. He has been a full professor at NTNU since 2010. His main research interests are performance and dependability modelling and simulations of communication networks, and now focusing on resource optimization and management in distributed, autonomous systems in a multi-domain context. He was the head of Department of Telematics (2009-13), and is now the head of the QUAM research lab at NTNU.

Table 1. Design space of DSI.

	DPI/DFI	Data Mining tools	DSI
Fine-grained analysis	✖	✓	✓
Flexibility	✖	✓	✓
Wire-speed	✓	✖	✓

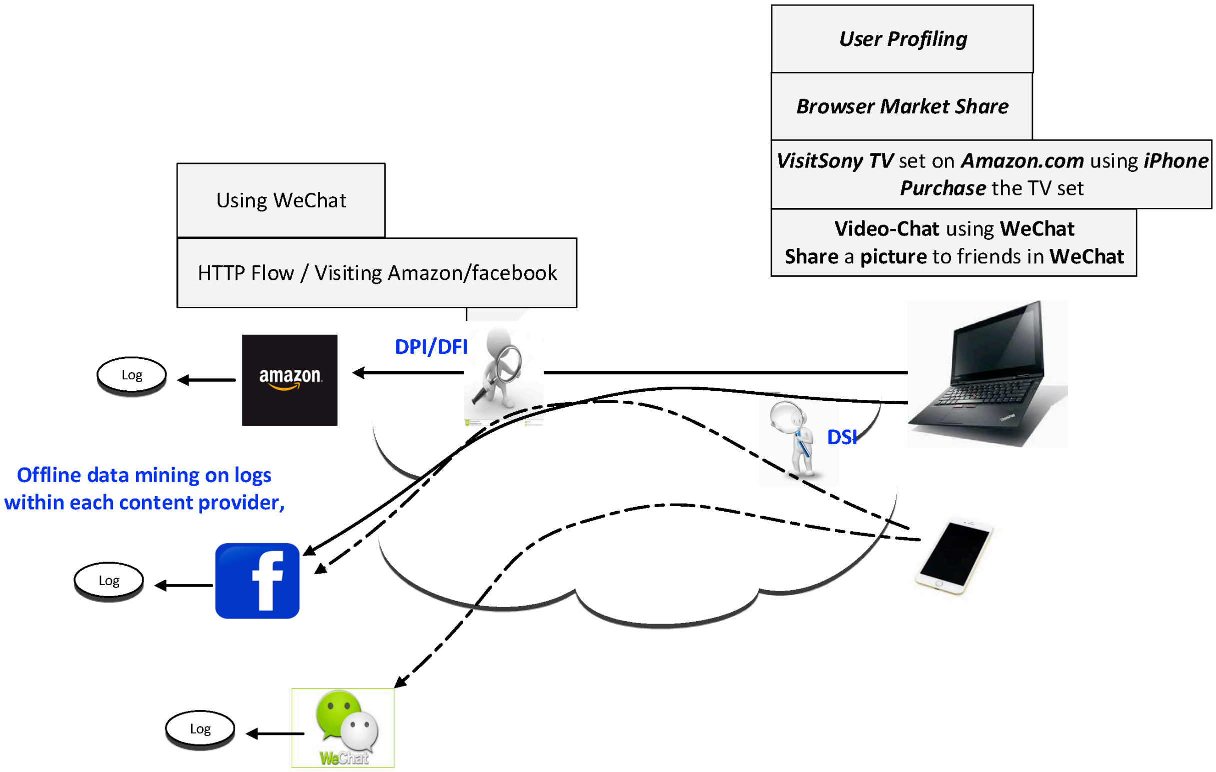


Figure 1. An illustrative example to compare DSI and DPI/DFI.

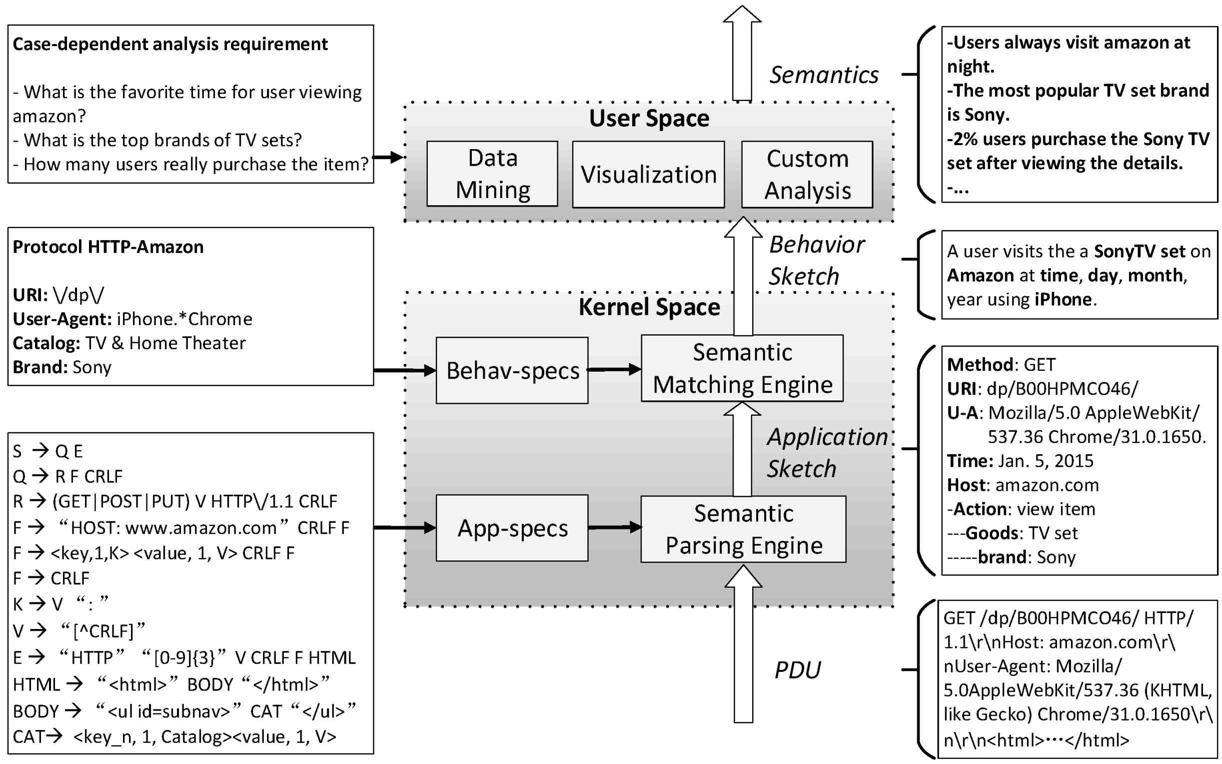


Figure 2. The system architecture of SOLID. The specifications, system architecture and the data flow, are shown in the left-hand, the middle, and the right-hand side, respectively.

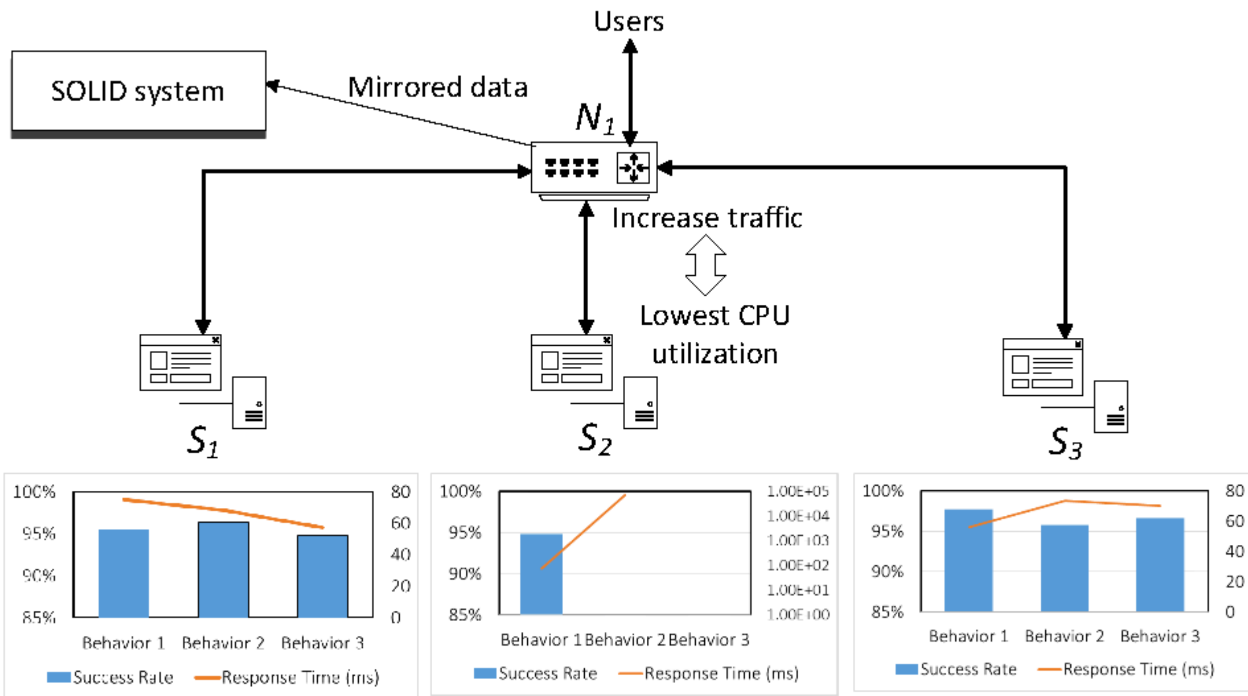
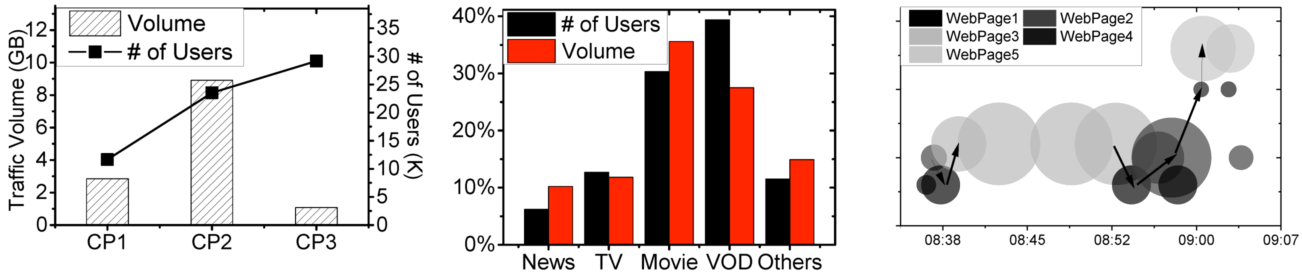


Figure 3. SOLID for the diagnosis purpose

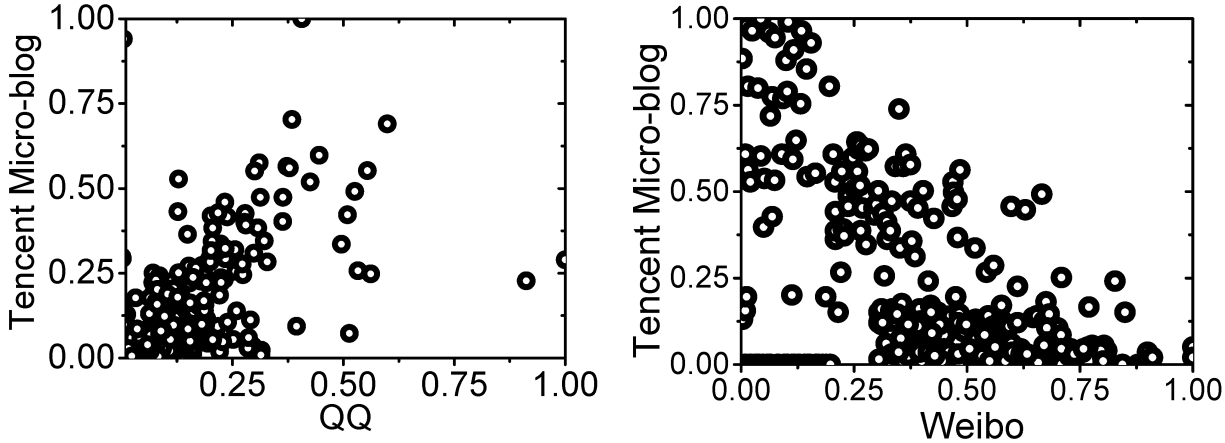


(a) The overview of #users and the traffic volume of the three CPs.

(b) The detailed analysis of CP3. Most users did not finish watching the videos.

(c) The clicking timeline for the "VOD" channel in CP3. Many users fall away on WebPage2.

Figure 4. The overview and detailed analysis of the three CPs with SOLID.



(a) The positive correlation between QQ and Tencent Microblog.

(b) The negative correlation between Weibo and Tencent Microblog.

Figure 5. The application correlation analysis.