# Deep Semantics Inspection over Big Network Data at Wire Speed

**Chengchen Hu, Hao Li, Yuming Jiang, Yu Cheng, and Poul Heegaard**

**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 of user behavior at wire speed, with a size several orders of magnitude smaller than that of 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.

Cisco Visual Networking Index report forecasted that global Internet traffic would 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], referred to as "big network data" in this article. If 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 a massive and rapidly increasing amount of data from possibly many different sources. The current solution 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, and so on is coarse-grained and inflexible, and will not give sufficient insight into users' behavior and preferences. In this article we describe an approach to gain deep understanding of users' behaviors and preferences extracted from protocol data units (PDUs) and the relationship between PDUs, and the agility to easily support various analysis purposes.

Specifically, we propose *deep semantics inspection* (DSI), where *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 descriptive meanings from the data at wire speed.

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

DSI discloses fine-grained information about the user: for the example in Fig. 1, after using Chrome on an iPhone to view a Sony TV set on Amazon on January 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, and browsers' market share. All these tasks can be completed at wire speed. This enhances the output from DPI, which only conducts protocol or application-level inspection (e.g., 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 analyses, but normally in an offline manner. In addition, DSI performs on 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.

## Inspecting Semantics over Big Network Data

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

*DSI should be operating at (close to) wire speed*. The wire speed in the core has been continuously increasing; 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 offline 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 the 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

*Chengchen Hu and Hao Li are with Xi'an Jiaotong University.*

*Yuming Jiang and Poul Heegaard are with the Norwegian University of Science and Technology.*

*Yu Cheng is with Illinois Institute of Technology.*

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. As one example, people use Facebook to share pictures, update timelines, chat with friends, and so on. As another example, users visit Amazon's web page, view product descriptions and other users' comments, compare prices, buy items, and so on. 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 desired 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 Fig. 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 to extract semantics from big network data in a fine-grained, flexible, and online manner.

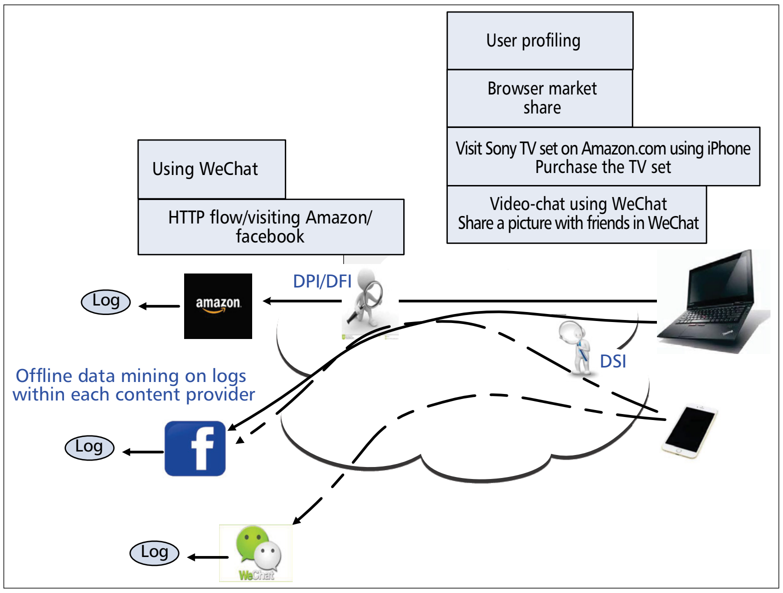

User profiling
Browser market share
Visit Sony TV set on Amazon.com using iPhone Purchase the TV set
Video-chat using WeChat Share a picture with friends in WeChat
Using WeChat
HTTP flow/visiting Amazon/ facebook
DPI/DFI
DSI
Log
Log
Log
Offline data mining on logs within each content provider

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

| | DPI/DFI | Data mining tools | DSI |
|---|---|---|---|
| Fine-grained analysis | × | ✓ | ✓ |
| Flexibility | × | ✓ | ✓ |
| Wire speed | ✓ | × | ✓ |

Table 1. Design space of DSI.

## 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 Fig. 2. For design details, please refer to [15].

### Data Flow of SOLID

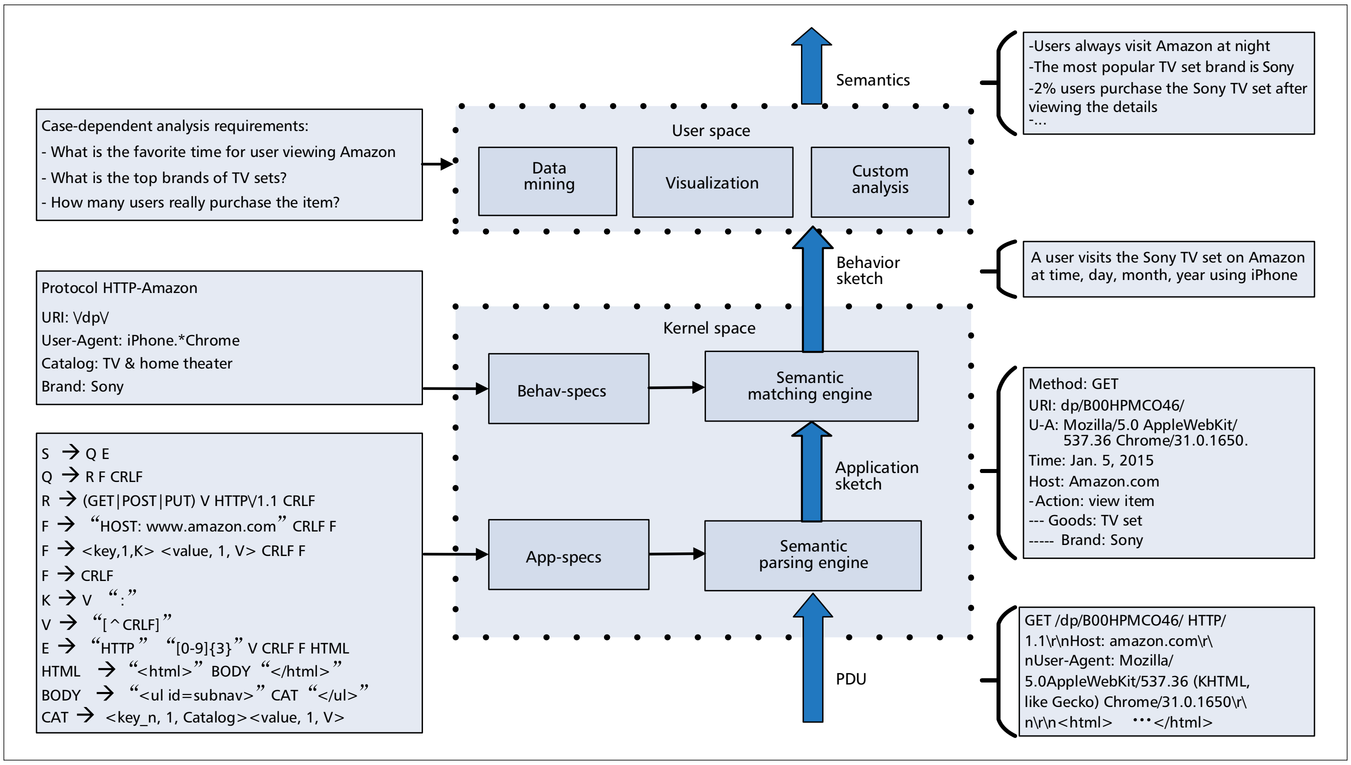
Basically, SOLID deduces the semantics over three main stages. It first transforms the raw PDU into an "application sketch" (app-sketch), which is a set of <field: value> pairs (**Time**: Jan. 5, 2015; **Host**: amazon.com; **Action**: view item, etc.). Next, SOLID works on the app-sketches to reveal 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 user behaviors. A detailed data flow example is shown on the right side of Fig. 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 aforementioned goals, respectively.

### Expressive Specification

We propose two specifications to express the applications and behaviors. First, app-spec is used to extract app-sketch, which is a predefined specification of the application protocols to parse a packet up to the application layer. It is generally in the form of a Backus-Naur form (BNF) with a set of production rules. The bottom left of Fig. 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 $E$. $Q$ further produces the request line $R$ and a set of header fields $F$. Following a similar process, we can eventually get a whole HTTP protocol with the interested information, such as the catalog of Amazon items. In addition, behav-spec is employed to extract the interesting properties from app-sketch, which is listed on the left side of Fig. 2 (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 layer 7 (L7) protocols, but for applications using their 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, the 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 used 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 an efficient handling process on the server side, applications tend to use structured format to carry their semantics (e.g., JSON/XML). These marked up languages are expressive and can be resolved easily.



Case-dependent analysis requirements:
- What is the favorite time for user viewing Amazon
- What is the top brands of TV sets?
- How many users really purchase the item?

```
Protocol HTTP-Amazon
URI: \/dp\/
User-Agent: iPhone.*Chrome
Catalog: TV & home theater
Brand: Sony
```

```
S → Q E
Q → R F CRLF
R → (GET|POST|PUT) V HTTP\/1.1 CRLF
F → "HOST: www.amazon.com" CRLF F
F → <key,1,K> <value, 1, V> CRLF F
F → CRLF
K → V ":"
V → "[^CRLF]"
E → "HTTP" "[0-9]{3}" V CRLF F HTML
HTML → "<html>" BODY "</html>"
BODY → "<ul id=subnav>" CAT "</ul>"
CAT → <key_n, 1, Catalog><value, 1, V>
```

-Users always visit Amazon at night
-The most popular TV set brand is Sony
-2% users purchase the Sony TV set after viewing the details
-...

A user visits the Sony TV set on Amazon at time, day, month, year using iPhone

```
Method: GET
URI: dp/B00HPMCO46/
U-A: Mozilla/5.0 AppleWebKit/
     537.36 Chrome/31.0.1650.
Time: Jan. 5, 2015
Host: Amazon.com
-Action: view item
--- Goods: TV set
----- Brand: Sony
```

```
GET /dp/B00HPMCO46/ HTTP/
1.1\r\nHost: amazon.com\r\
nUser-Agent: Mozilla/
5.0AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/31.0.1650\r\
n\r\n<html> ···</html>
```

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

## Agile User Space

The architectural design of SOLID is illustrated in the middle column of Fig. 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 applicable 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 application programming interfaces (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 in an agile way. The user space of SOLID provides the flexibility for different applications to produce their own specific semantics based on the unique framework of SOLID, as well as the behavior sketch input. In the next section, we present three practical scenarios to demonstrate the potential of the user space.

## Fast Kernel Space

The design of the kernel space determines the performance of SOLID in achieving the wire speed processing goal.

The bottom layer in the SOLID architecture, the semantics parsing engine (SPE), resolves the reassembled PDUs into the application sketch according to the app-specs. The 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 ensure the high-speed processing of the SPE. Please refer to our previous work [7] for the detailed design of the parsing method. Previous works are not sufficient for our purpose. For example, Binpac [8] extracts "http-request," "http-request-header," and "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, and compares the application sketch with the predefined behav-spec and outputs the behavior sketch. The DSI system is expected to scale with emerging specifications resulting from the growth of new applications/functions. We 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 possibly be 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 later extended ROOM to MP-ROOM [11] to support multiple PDUs for more complex behav-specs, which is used as the SME of SOLID. An intrusion detection system (IDS) [12] also has a matching component to detect intrusions, but it 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. Although 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

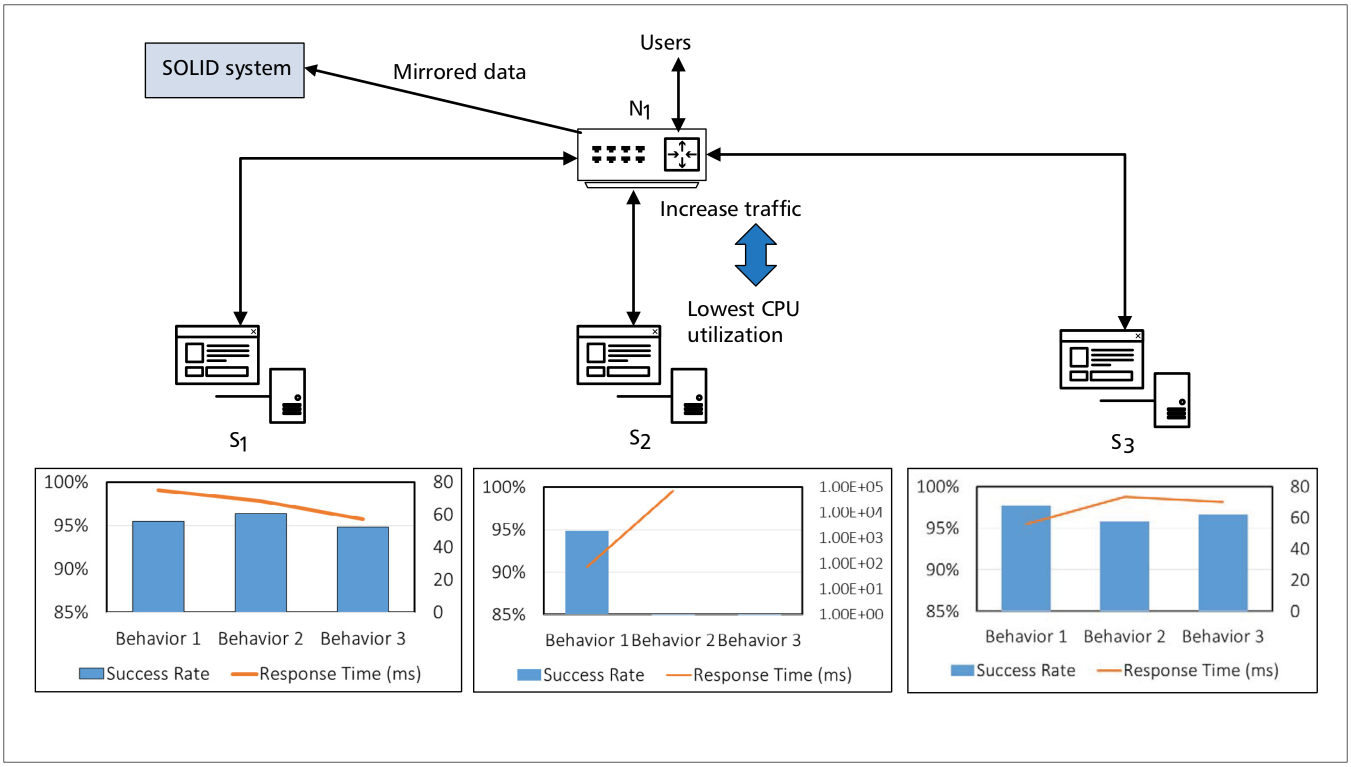


Figure 3. SOLID for diagnostic purposes.

every flow should have a match in SOLID, increasing the processing pressure of the SME.

## Practical Cases

SOLID has been deployed in several practical scenarios, and three representative ones are presented below. These cases are 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, respectively. 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 diagnosis. 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 input 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.

One day during the deployment of SOLID, $S_2$ failed to respond the correct data due to a disk error. $S_2$ could not provide any service, but $N_1$ kept on dispatching new requests to $S_2$ due to its low CPU utilization. Traditional diagnostic 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 in the bottom part of Fig. 3, SOLID monitored three behaviors in the servers. Normal nodes $S_1$ and $S_3$ experienced large success rates and small response times, while $S_2$ failed to respond to behavior 2 and did not trigger the request of behavior 3.

### Consulting Analysis

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

SOLID is able 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 detail. The output semantics in this case is the competition analysis based on the statistics of behavior sketches.

Figure 4a 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 in attracting users, but the low traffic raises a potential problem, as does how to make users stick to it. To understand the problem better, Fig. 4b shows the statistics of CP3 in detail]. The "VOD" channel attracts most of the users, but does not produce very much traffic. In other words, users do not pay much time or money on watching the whole video but just glance at them. An implication is that there could be a risk of losing users if CP3 cannot provide interesting/attractive content. In addition, Fig. 4c illustrates the users' clicking pattern in CP3's VOD channel. Each circle is a web page, and its diameter is proportional to the page views on this page. We could infer many suggestions from this graph: 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 on WebPage2, which lowers the traffic on WebPage4. Usually, CP can perform such analysis with the web logs individually, but SOLID can do such mining at the

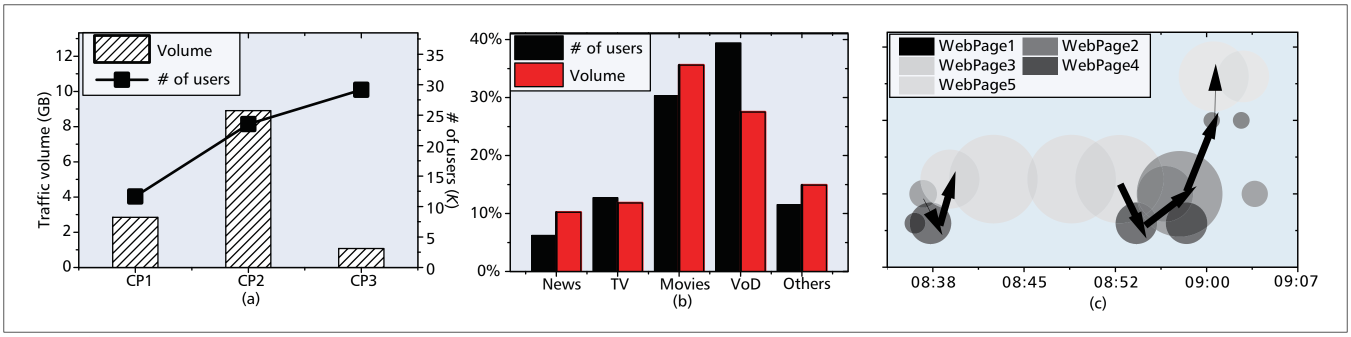


Figure 4. The overview and detailed analysis of the three CPs with SOLID: a) the overview of users and the traffic volume of the three CPs; b) detailed analysis of CP3 (most users did not finish watching the videos); c) the clicking timeline for the VOD channel in CP3. Many users fall away on WebPage2.

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 offline analysis on the weblogs from a single CP.

### 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 news to its users and Twitter as a social network service (SNS) platform supporting one-to-one chatting. On the other hand, some applications are complementary: a news feed application can share 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.

We demonstrate one-day frequencies of several popular applications in an NSP's network and evaluate their correlations in Fig. 5, where the X and Y axes are the normalized frequencies of using the corresponding applications. Figure 5a shows the positive correlation between QQ and Tencent Microblog. Since QQ has embedded some functions of Tencent Microblog, Tencent Microblog can gain popularity from QQ. Figure 5b 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.

## Performance Evaluation

Although SOLID outputs consistent and solid results, as 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 32 GB memory. We preload two real traces with their original segment orders. One was captured from the campus network of a university in China obtaining 4.5 GB traffic. The other 7.5 GB 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, and online shopping. We further give 1048 behavior specifications for the evaluations.

The single thread implementation of SOLID achieves 3.0 Gb/s and 2.7 Gb/s with the two real traces, respectively (i.e., about 1.7 times faster than NetShield [12]). It is reported in [12] that NetShield reaches an analysis speed of 11 Gb/s on a DARPA trace, while the throughput of SOLID on the same trace is 16.9 Gb/s. With multiple-thread evaluation using 10 cores, SOLID's throughput is 17.2 Gb/s for the first real trace and 15.9 Gb/s for the second. Considering that the throughput of NetShield is measured without reassembly work and only for HTTP with fewer (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 708 MB and 839 MB during the test with different 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 growing rate gap over the big network data.

## Discussions

This article introduces the concept of deep semantics inspection, 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 aforementioned principle. It is highly desired to study the (semi-) automation of this process, especially when 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 article 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 an 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 a 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 minimized but descriptive enough sketch categories, like who, when, where, subjects, objectives, actions, and so on, would be feasible.

## 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 user intent by gathering unstructured data into a unified framework. Three real cases leverag-

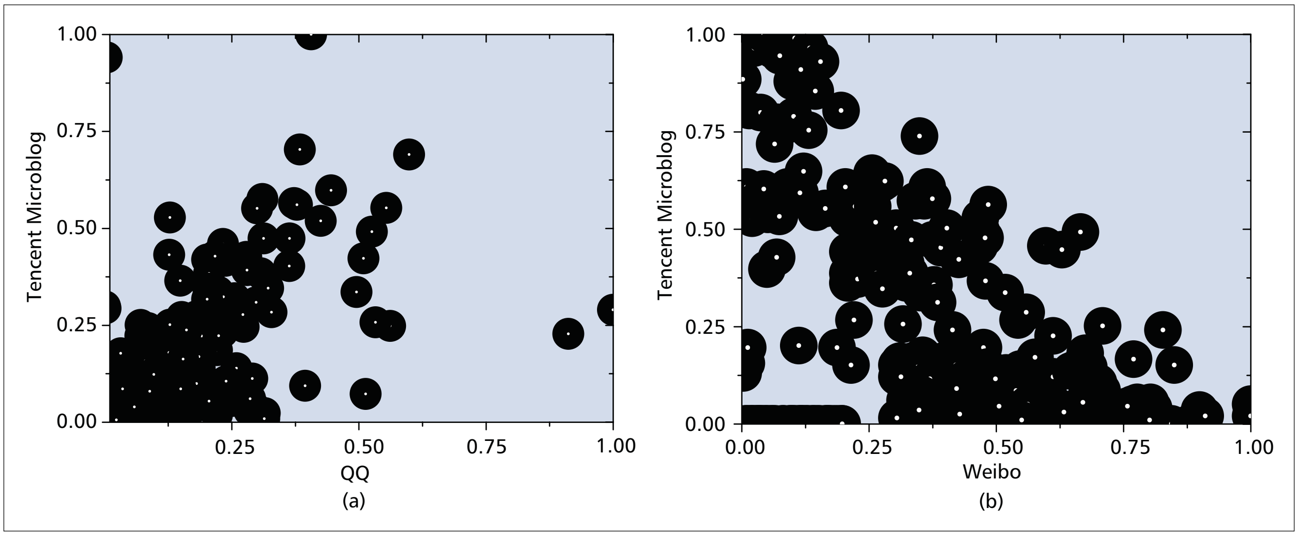


Figure 5. The application correlation analysis: a) The positive correlation between QQ and Tencent Microblog: b) the negative correlation between Weibo and Tencent Microblog.

ing DSI/SOLID, as well as performance experiments on high throughput and efficient memory usage, have demonstrated the usage and feasibility of DSI.

## Acknowledgments

This article 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"; 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 *et al.*, "MicroTE: Fine Grained Traffic Engineering for Data Centers," *CoNEXT*, New York, NY, 2011, pp. 8:1–8:12.

[4] "Market Share for Mobile, Browsers, Operating Systems and Search Engines," 2013; http://www.netmarketshare.com/.

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

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

[7] H. Li *et al.*, "Parsing Application Layer Protocol with Commodity Hardware for SDN," *Proc. 11th ACM/IEEE Symp. Architectures for Networking and Commun. Systems*, Oakland, CA, 2015, pp. 51–61.

[8] R. Pang *et al.*, "binpac: A yacc for Writing Application Protocol Parsers," *Proc. 6th ACM SIGCOMM Conf. Internet Measurement*, 2006, pp. 289–300.

[9] C. Meiners et al., "Flowsifter: A Counting Automata Approach to Layer 7 Field Extraction for Deep Flow Inspection," *2012 Proc. IEEE INFOCOM*, 2012, pp. 1746–54.

[10] H. Li and C. Hu, "ROOM: Rule Organized Optimal Matching for Fine-Grained Traffic Identification," *2013 Proc. IEEE INFOCOM*, 2013, pp. 65–69.

[11] H. Li and C. Hu, "MP-ROOM: Optimal Matching on Multiple PDUs for Fine-Grained Traffic Identification," *IEEE JSAC*, vol. 32, no. 10, Oct. 2014, pp. 1881–93.

[12] Z. Li *et al.*, "NetShield: Massive Semantics-Based Vulnerability Signature Matching for High-Speed Networks," *Proc. ACM SIGCOMM 2010 Conf.*, New York, NY, 2010, pp. 279–90.

[13] Y. Wang *et al.*, "A Semantics Aware Approach to Automated Reverse Engineering Unknown Protocols," *20th IEEE Int'l. Conf. Network Protocols*, 2012.

[14] J. Sherry *et al.*, "BlindBox: Deep Packet Inspection over Encrypted Traffic," *Proc. ACM SIGCOMM*, 2015.

[15] C. Hu *et al.*, tech. report, "Deep Semantic Inspection for On-Line Analysis of Big Network Data," Xi'an Jiaotong Univ., 2015; 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, and software defined networking. He is the recipient of a fellowship from the European Research Consortium for Informatics and Mathematics (ERCIM), New Century Excellent Talents in University awarded by the Ministry of Education, China.

HAO LI [S'13] received his B.S. degree in computer science from Xi'an Jiaotong University in 2010, and is now a Ph.D. candidate in the Department of Computer Science and Technology at the same 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 his Ph.D. degree from National University of Singapore. Since 2005, he has been a professor with the 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 the IFIP Networking 2014 Conference and is the author of *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 Electrical and Communications Engineering Department, Illinois Institute of Technology. His research interests include wireless networks, network security, and next-generation Internet technology. He has 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 Ph.D. in telematics from NTNU in 1998. He has been a full professor at the same university since 2010. His main research interests are performance and dependability modelling and simulations of communication networks, currently focusing on resource optimization and management in distributed autonomous systems in a multi-domain context. He was head of the Department of Telematics (2009–2013), and is now head of the QUAM research lab at NTNU.