# A Survey of Distributed Event Systems: Post-print

**Authors:** Hu Songlin

**Date:** 2017-03-10T00:00:00+00:00

## Abstract

This paper briefly reviews the concept, evolution, and main research topics of distributed event systems, and introduces our research and application work in this area.

## Full Text

## Preamble

Vol. 9 No. 5
Information Technology Letter

## Survey of Distributed Event-Based Systems Research

**Abstract:** This paper provides a brief overview of the concepts, evolution, and main research topics of distributed event-based systems, and introduces our research and application work in this area.

**Keywords:** Distributed event-based systems; publish/subscribe; content-based routing; complex event processing

## 1 Introduction

Events are ubiquitous in human activities, varying widely in granularity and type. An event can be a phenomenon or news story in daily life, a real-time quote in financial trading, a production order, or a message triggered by a mouse movement in a computer system. Whether in society, daily life, or computer systems, diverse events are continuously generated, driving both society and individuals—much like computers—to participate in cyclical event processing activities.

The purpose of Distributed Event-Based Systems (DEBS) is to efficiently filter or process information events provided by producers and deliver them to interested consumers, creating an event-driven, loosely coupled distributed environment. DEBS are widely applied in real-time processing, system monitoring, continuous query, business process management, and other domains. Here, information producers are typically called "Publishers," while consumers are called "Subscribers," connected through a broker network. In practical applications, publishers can be application components, database triggers, sensor devices, smart grid terminals, or multimedia content providers, while subscribers can be real-time information query and analysis tools, application components, device controllers, databases, etc. Subscribers describe their interests by submitting subscriptions, and DEBS broker nodes compute matches between publications and subscriptions at the channel, topic, or content level, then selectively route events to interested subscribers based on matching results.

Early event processing researchers came from diverse fields including databases, distributed systems, middleware, and software engineering. Event processing technology with content-level filtering and processing capabilities attracted widespread attention, serving as a catalyst for converging various streams and forming an independent community. This content-based distributed event processing research originated in Europe and gradually matured over more than a decade of development. In 1998, Antonio Carzaniga from Politecnico di Milano published his seminal PhD dissertation, "Architectures for an Event Notification Service Scalable to Wide-area Networks," systematically articulating the models, algorithms, and implementation of content-based distributed event notification systems, marking the beginning of this research field. In 2002, Professor David Luckham' s book *The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems* brought complex event processing technology into the purview of event system researchers. In 2003, Carzaniga and Wolf first proposed the concept of "content-based network" at SIGCOMM [1], further expanding the development space of DEBS into the networking domain.

To advance research in DEBS and provide a high-level communication platform for researchers, since 2002, under the advocacy of Professor Hans Arno Jacobsen from the University of Toronto, the DEBS research community has actively organized five workshops at three top-tier international conferences across different domains: SIGMOD, ICDCS, and ICSE. While focusing on matching and routing algorithms, researchers also addressed issues such as event-driven architecture and service computing, discussed teaching systems and curriculum design for DEBS, and began forming a relatively independent scientific community. In 2007, Gartner released a report [2] declaring that event processing would bring breakthrough innovation and become the "Next Big Thing," leading to growing recognition of the importance of events. In September 2007, Gartner hosted the first Event Summit, attracting broad participation from the DEBS academic community and major companies including IBM, Oracle, Microsoft, and CA. In October, the first ACM Conference on Distributed Event-Based

Systems was held in Toronto, gathering renowned scholars and industry leaders in the field and marking the formal establishment of the research community [3]. During this period, DEBS technology applications also entered an accelerated development phase, expanding from traditional information dissemination [4], process management/enterprise application integration [5], and data stream management [6] to emerging domains such as blog filtering [7], ad hoc networks [8], cloud computing [9], and smart grids [10].

## 2 Main Research Content

A prominent challenge facing DEBS is the "large-scale"characteristic of the entire system. "Large-scale" here primarily refers to: (1) extremely large numbers of concurrent events; (2) large numbers of active subscriptions that can be updated autonomously as subscriber interests change; (3) large populations of publishers or subscribers; and (4) numerous network nodes that may be geographically distributed across vast areas.

In such a large-scale network environment, event matching/filtering and routing become two fundamental problems for achieving on-demand, efficient, and reliable delivery of massive events. In recent years, technologies essential for usable distributed systems—such as load balancing, fault tolerance, mobility, and security—as well as application research driven by novel applications have gradually become new hotspots in DEBS research.

### 2.1 Event Matching/Filtering

Event matching/filtering can be simply categorized into three types based on filtering capability: channel-based filtering, topic-based filtering, and content-based filtering.

In channel-based matching, publishers and subscribers each select a corresponding channel identifier (ID), and event filtering is accomplished through direct ID matching. In topic/subject-based matching, both publishers and subscribers use a string or a path in a type tree to describe published content and subscription interests, with matching determined through string matching or equality/containment relationships between types.

Compared to the former two, content-based filtering relies entirely on event content for matching, offering more powerful descriptive and information processing capabilities and representing the focus of DEBS research. Such event systems can be simply viewed as inverted databases: queries (subscriptions) are relatively static while data (publications) change dynamically. The purpose of event matching or filtering is to continuously identify event sets that satisfy a group of subscriptions over ongoing event streams. Generally, event models supporting content matching and filtering include attribute/value pair models, XML models, and RDF models, with different models requiring different algorithmic approaches.

**Attribute/Value Pair Model.** The attribute/value pair model is the most commonly used model, describing events as a set of attributes and their values. Typical systems such as Gryphon [33] and PADRES [34] adopt this model. In this model, a publication P is a set of "attribute-value" pairs generated by information producers. Each attribute filter iF (where $0 < i < n$, with n being a natural number representing the number of filter conditions) is a predicate of the form "attribute-operator-value." A subscription can be defined as a series of attribute filters. If a publication P satisfies all filter conditions of subscription S, then P matches S.

Classic algorithms for attribute/value pair matching include: the brute-force algorithm (set enumeration), counting algorithm, decision tree algorithm, binary decision diagram algorithm, and two-phase matching algorithm.

The brute-force algorithm simply enumerates all subscriptions against notification messages. If all conditions of a subscription are satisfied, it returns true; otherwise, false. While simple and direct, this approach leads to redundant predicate computations, affecting overall system efficiency.

To address these limitations, Tak W. Yan [11] proposed a counting algorithm. In this approach, each predicate points to multiple filters, with each filter containing a counter initialized to 0 and reset after each event matching operation. Any given event is only matched against all predicates. When a predicate is matched, the counters of corresponding filters are incremented. When a filter' s counter value equals the total number of its constituent predicates, the match is declared successful. Yan also established a three-layer index for predicate attributes, operators, and values to further improve algorithmic efficiency.

M. K. Aguilera et al. [12] proposed a decision tree algorithm that uses filters as leaf nodes, "attributes and operators" as non-leaf nodes, and "values" as edges. The decision process involves traversing the tree from the root node, with leaf nodes reached by satisfying node and edge conditions returning true, and others returning false.

The counting and decision tree algorithms are designed for filters composed of predicates with "AND" relationships and cannot handle filters containing "OR" relationships. To address this, A. Campailla et al. [13] proposed a matching algorithm based on Binary Decision Diagrams (BDDs). This structure includes two terminal nodes, 1 and 0, with each predicate forming a non-terminal node containing two outgoing edges: a high edge indicating predicate satisfaction and a low edge indicating non-satisfaction. An edge points to the next predicate or a terminal node. They also proposed Ordered Binary Decision Diagrams (OBDDs) to enable sharing among multiple filters.

The two-phase matching algorithm is an efficient optimization [14] that operates in two stages: first, identifying all subscription subsets with at least one attribute matching the publication; second, using the counting approach to calculate how many conditions of these subscriptions are satisfied. If all conditions are met, the subscription match returns true.

**XML/RDF Model.** The XML model uses XML to describe publications and XPath/XQuery to describe subscriptions. M. Altinel and M. J. Franklin [15] proposed XFilter for XML information dissemination, representing XPath expressions as Finite State Machines (FSMs). When conditions in an XML document continuously trigger an FSM to reach an accepting state, the XML document is considered to match the corresponding XPath. However, since all XPath expressions are independent, XFilter suffers from similar defects as the brute-force algorithm. To address this, Yanlei Diao, together with the XFilter authors, proposed YFilter [16], which merges all XPath expressions into a single Nondeterministic Finite Automaton (NFA) to share common filter conditions among different expressions, thereby reducing matching operations. Additionally, Guoli Li [17] proposed a method using attribute/value pairs for XML description and matching to effectively improve system matching speed.

M. Petrovic, addressing content-based RSS (RDF Site Summary) dissemination requirements, used graph models to describe RDF-formatted publications and subscriptions, proposing a G-ToPSS model [18] comprising publication Gp, subscription Gs, and ontology. Event matching is thus transformed into a graph matching problem: if Gs matches a subgraph of Gp, then Gp matches Gs. Petrovic also proposed a two-layer hash table structure to improve system efficiency.

**Extended Models.** *Advertisement-based Model.* In simple subscription-based models, subscription information needs to be flooded throughout the network, often incurring enormous communication overhead. Considering that the number of information publishers is typically much smaller than subscribers, advertisement-based models [19] have been proposed, where the smaller number of advertisements are flooded instead. An advertisement can be viewed as a "notice" from publishers, containing a series of attribute filters describing the value range for each attribute in the information they publish. Any advertisement A flooded to the network first matches subscriptions, obtaining a subset of subscriptions matching the advertisement. The corresponding publication P from advertisement A then only needs to match against this subset, reducing matching computation and improving efficiency.

*Composite Event/Subscription Model.* Composite events are sets of events with relationships. Reference [20] defines basic types such as event concurrency, co-occurrence within time windows, selection relationships, and sequential relationships, proposing a modeling method based on finite automata. Detection and matching of such events can be solved using finite automata theory. In advertisement-based models, Guoli Li [21] proposed composite subscriptions, which define requirements for events as Boolean operations among multiple atomic subscriptions, expressed as a binary tree structure and implemented through matching individual subscriptions with events and solving the binary tree to fulfill composite event subscription requirements. Compared to composite events, composite subscriptions only describe the compositional relationships among desired events, focusing research on routing algorithm optimization, whereas composite events more finely distinguish relationship types such as co-

occurrence within time windows, concurrency, and sequencing, offering stronger expressive power.

### 2.2 Event Routing

From a network structure perspective, DEBS networks include acyclic overlay and cyclic overlay types. Routing algorithms for acyclic networks are relatively mature, while those for cyclic networks have recently become a focus of DEBS research. Based on routing table structure, DEBS topologies include two types: unstructured overlay networks and structured peer-to-peer (P2P) networks (i.e., Distributed Hash Table-based networks). The former adopts a topology consistent with Gnutella networks, using classical routing algorithms, while the latter employs DHT-based routing methods, often combined with rendezvous node routing in implementation. DEBS networks can also combine these structures to form hierarchical or hybrid architectures to meet different application requirements.

Common routing algorithms include the following five types:

**Subscription-based Routing.** As shown in [Figure 1: see original paper], this model first floods all subscriptions (e.g., S1 and S2 in the figure) throughout the network. When a new publication (Pub) is generated, it matches corresponding subscriptions (Sub) at broker nodes, then forwards the publication step-by-step along the reverse path of the subscription to the broker connected to the subscriber. For example, in the figure, event n1 generated by publisher P satisfies S2' s condition v>2, so n1 is forwarded to S2. (For simplicity, we assume all subscriptions and publications have only one attribute v.) To ensure newly created subscriptions match publications already being routed in the network, existing DEBS implementations typically perform new matching operations at all brokers along the publication' s path to avoid missing new subscriptions.

**Advertisement-based Routing.** As shown in [Figure 2: see original paper], in advertisement-based systems, publishers must first send advertisement messages A to reach every broker node in the network before publishing messages. If a broker node has subscription information matching the advertisement content, the subscriber' s subscription information (e.g., S2 in the figure) is returned step-by-step against the advertisement propagation direction to the node directly connected to the publisher, forming a "subscription routing tree." Since S1' s value v>3 does not satisfy advertisement A' s conditions, S1 does not form a subscription routing tree for A. Thus, when the publisher releases new information (e.g., n1) that satisfies subscription constraints, the publication backtracks along the subscription routing tree (e.g., S2' s routing tree) from leaf nodes to the broker connected to the subscriber, which then sends the message to the subscriber.

**Cyclic Network Routing.** Carzaniga [23] proposed a content-based routing strategy supporting cyclic network topologies. Its advantage lies in using distance vector methods to establish subscription paths, enabling events to be sent

to subscribers via shortest paths. However, this approach discards duplicate messages rather than preventing their generation from the start, creating numerous message copies. Additionally, for subscriptions on cycles, they may not forward matching events toward covered subscription directions if that direction is not the shortest path for the covered subscription. Guoli Li [24] proposed a new content-based cyclic network routing protocol called ECBR. It employs a scheme with advertisement filters and assigns identifiers to subscription filters, with events avoiding duplicate message transmission and errors on loops through ID-based path routing. Since subscribers in ECBR establish multiple subscription paths based on advertisement trees, events at the intersection of these paths can use heuristic methods based on current network conditions to find optimal paths to subscribers, effectively improving routing efficiency.

**Rendezvous Point Routing.** In rendezvous point routing structures, multiple rendezvous points exist for different event types, with one rendezvous point corresponding to one or multiple event types. Broker nodes matching these types, together with the rendezvous point, form an independent distribution subgraph. The entire network can also achieve load balancing by distributing event types across multiple rendezvous points. In this distribution subgraph, subscriptions do not require broadcasting. As shown in [Figure 3: see original paper], subscriptions S1 and S2 first route through the broker network until reaching a broker directly connected to rendezvous point R, then are forwarded to R. If publication n1 from publisher P1 encounters matching subscriptions while en route to R, it is sent directly to subscribers along the reverse path of the subscription. For example, n1 is sent to S1 in the figure. After reaching rendezvous point R, it is forwarded to other brokers containing subscriptions satisfying n1' s conditions and ultimately sent to corresponding subscribers, such as S2 in the figure. Since ordinary broker nodes in the network need to locate rendezvous points for specific event types, DHT-based methods are often used for rendezvous point discovery.

**Composite Event/Subscription Routing.** Composite event routing and composite subscription routing share similarities. The composite subscription routing proposed in [21] is representative. When subscribers submit composite subscriptions, brokers decompose them step-by-step based on how atomic subscriptions in the composite match advertisements, then route them separately. As shown in [Figure 4: see original paper], a composite subscription composed of atomic subscriptions S1, S2, and S3 splits out S3 at Broker 2 and S1 and S2 at Broker 4. This approach effectively reduces forwarding overhead for composite subscriptions.

### 2.2.1 Routing Optimization

**Covering-based Routing.** If subscription S1' s filter conditions contain S2' s, and S1's subscription path contains S2's, then S1 can represent S2's subscription interest—that is, S1 covers S2. As shown in [Figure 5: see original paper], S1' s v value is greater than 1, and S2's v value is greater than 2, so S1 covers S2. In this

case, at the intersection point in the figure, we only need to continue forwarding S1, not S2. On the shared path, for any matching message (e.g., n1(v=3) in the figure), S1 receives messages on behalf of S2. This effectively reduces subscription forwarding message volume, compresses routing table space, and improves matching efficiency.

**Merge-based Routing.** Merge-based routing shares some similarities with covering-based routing [25]. Its basic idea is to find the union of filter conditions from a set of subscriptions {Si} and generate a new subscription Sm to serve as a proxy for {Si}. Thus, on any overlapping path between Sm and any subscription in {Si}, {Si} can stop forwarding. As shown in [Figure 6: see original paper], if S1' s v value belongs to [4,6] and S2' s v value belongs to [3,5], we can generate a union Sm: v belongs to [3,6] to replace S1 and S2 routing. However, computing unions on complex subscription sets with multiple attribute-value pairs is costly, and determining the optimal size of subscription groups for merging to achieve the best effect is also challenging. Additionally, merge-based routing can sometimes cause spam: publication events matching the merged Sm but matching no individual subscription in the group are still routed to Sm' s starting point, incurring unnecessary message transmission overhead. Therefore, merge-based routing is less commonly used than covering-based routing.

**Interest Clustering.** If similar subscriptions share paths, covering-based routing optimization can significantly reduce publication message volume. Based on this characteristic, references [26][27] studied subscriber clustering methods. The basic idea is to divide system subscribers into several groups based on subscription interests [27], then regroup these subscriptions and distribute them as evenly as possible to nearby brokers to improve covering routing optimization effectiveness.

### 2.3 Other Research Topics

Building upon composite events, some researchers have further considered more complex correlation relationships across multiple event streams and the need for more powerful processing capabilities, leading to in-depth research on complex event processing technology. From a system perspective, complex event processing incorporates operators for event aggregation, splitting, and simple computation, combined with concepts from continuous queries in databases, introducing new time- or length-based windows such as "sliding windows" to better meet practical application requirements. The team that proposed the XML event matching algorithm YFilter, led by Yanlei Diao, has conducted a series of influential studies on complex event processing models, languages, and algorithms [28][29]. Other scholars have contributed to RFID applications [30] and optimization in distributed environments [31]. Other DEBS research topics include hardware acceleration of algorithms, security, fault tolerance, transaction processing, load balancing, and more.

## 3 Our Research and Applications

### 3.1 Main Research Work

Focusing on core DEBS technologies and application models, we have conducted multifaceted research, publishing a series of full-length academic papers at top-tier or important conferences in software engineering and distributed systems such as OOPSLA, ICDCS, and DEBS.

In matching algorithms, we proposed event matching and routing methods based on social and conceptual integration networks, applying them to question routing in online programming forums to effectively improve question matching efficiency. The related paper became the second article from mainland China published at OOPSLA after IBM China Research Laboratory. In routing optimization, addressing the lack of covering-based routing methods for cyclic networks, we proposed a covering routing protocol for cyclic networks with formal verification; we also identified a serious flaw in covering optimization algorithms that can generate massive instantaneous messages when canceling covering relationships, and proposed a selective covering cancellation routing algorithm that significantly improves cancellation efficiency and resolves system bottlenecks. We also investigated load balancing and task migration issues in distributed systems, focusing on client reliable migration protocols, routing table reconfiguration optimization algorithms, and optimized deployment methods for DEBS. We applied DEBS technology to Service-Oriented Architecture (SOA) research, proposing event-based distributed resource/service discovery and composition methods that effectively improve operational efficiency. Applying some algorithmic ideas to massive service processing, we won first place with perfect scores across all five categories and fifteen items in the 2009 International Web Services Competition and successfully defended the title in 2010.

### 3.2 Smart Grid Applications

Real-time event processing for smart grids represents a new growth point for DEBS applications and is a key application focus of our research group. With the rapid advancement of smart grid construction, smart meters, distributed clean energy, electric vehicles, and smart appliances have been widely deployed. At the grid' s edge in the electricity consumption segment, massive new terminal devices and bidirectional interaction between devices and the grid have generated enormous volumes of real-time, high-frequency grid events. A single provincial pilot project in China involves tens of millions of terminals and hundreds of thousands of broker nodes, with event processing reaching hundreds of millions per collection cycle. To address these challenges, we collaborated with the State Grid Information & Telecommunication Company (the contractor for the Smart Grid Pavilion at the World Expo) to develop the LIT (fLexible hIgh Throughput event Processing platform for Smart Grid) high-throughput event processing platform based on our early research prototype, combined with the company' s extensive smart community construction experience and urgent

requirements, and began pilot applications.

LIT, meaning "ignite," "illuminate," or "hold high," aligns with the State Grid Pavilion' s Expo slogan "Innovation Illuminates Dreams." LIT supports flexible access for massive smart grid device endpoints, providing a networked environment for large-scale event filtering, efficient routing, and distributed storage, while supporting online analytical processing of complex events. It provides common foundational support for information processing technologies throughout the entire lifecycle of massive, highly concurrent events, including access, transmission, storage, analysis, monitoring, response, and control.

## 4 Summary and Outlook

After several years of development, the DEBS field has initially formed an active academic research community, a preliminary theoretical system, and experimental applications from various related fields are being carried out vigorously, creating a promising situation. Current foundational research work not only lays a solid foundation for future large-scale application of DEBS technology but has also provided and will continue to provide many algorithms and protocols of reference value for other related disciplines.

Although DEBS technology research has begun to take shape, its core key technologies and application models remain under continuous exploration. The industry generally believes we are still at a stage of "not fully understanding" DEBS [32]. Against this background, researching core algorithms and novel application models in DEBS holds significant forward-looking research and application value.

## References

[1]. Carzaniga, A. and Wolf, A. L. 2003. Forwarding in a content-based network. SIGCOMM 2003. Karlsruhe, Germany: 163-174.
[2]. Massimo Pezzini, Yefim V. Natis. 2007. Trends in Platform Middleware: Disruption Is in Sight, Gartner Report.
[3]. Peter Pietzuch, Gero Muhl, Ludger Fiege. 2007. Distributed Event-Based Systems: An Emerging Community, DEBS2007. Toronto, CA: 2-
[4]. Pardo-Castellote, G., 2003. OMG Data-Distribution Service: architectural overview, In Proceedings of ICDCS Workshops, 2003. Providence, Rhode Island, USA: 200-206,
[5]. Guoli Li, Vinod Muthusamy, and Hans-Arno Jacobsen. January 2010. A distributed service-oriented architecture for business process execution. ACM Trans. Web 4, 1: 1-33
[6]. Vibhore Kumar, ZhongTang Cai, Brain F. Cooper, etc. 2006. IFLOW: Resource-aware Overlays for Composing and Managing Distributed Information Flows, EuroSys 2006. Leuven, Belgium
[7]. I. Rose, R. Murty, P. Pietzuch, J. Ledlie, M. Roussopoulos, and M. Welsh. 2007. Cobra: Content-based Filtering and Aggregation of Blogs and RSS Feeds.

In Proceedings of the 4th USENIX Symposium on Networked Systems Design & Implementation. Berkeley, CA, USA: 29 42.
[8]. L. Mottola, G. Cugola, and G. P. Picco. Aug. 2008. A Self-Repairing Tree Topology Enabling Content-Based Routing in Mobile Ad Hoc Networks. IEEE Transactions on Mobile Computing, 7(8):946 960.
[9]. Brian F. Cooper, Raghu Ramakrishnan, Utkarsh Srivastava, Adam Silberstein, Philip Bohannon, Hans-Arno Jacobsen, Nick Puz, Daniel Weaver, and Ramana Yerneni. 2008. PNUTS: Yahoo!' s hosted data serving platform. Proc. VLDB Endow. 1, 2 (August 2008): 1277-1288
[10]. Sood, V.K, Fischer D, Eklund J.M, etc. 2009. Developing a communication infrastructure for the Smart Grid. IEEE 2009 Electrical Power & Energy Conference. Montreal, Quebec, Canada: 1-7.
[11]. Yan T.W., Garcia-Molina, H. 1994. Index structures for information filtering under the vector space model, ICDE. Houston, Texas: .337-347.
[12]. Aguilera, M. K., Strom, R. E., Sturman, D. C., Astley, M., and Chandra, T. D. 1999. Matching events in a content-based subscription system. PODC 1999. New York, NY: 53-61.
[13]. Campailla, A., Chaki, S., Clarke, E., Jha, S., and Veith, H. 2001. Efficient filtering in publish-subscribe systems using binary decision diagrams. ICSE 2001, Washington, DC: 443-452.
[14]. Fabret, F., Jacobsen, H. A., Llirbat, F., Pereira, J., Ross, K. A., and Shasha, D. 2001. Filtering algorithms and implementation for very fast publish/subscribe systems. SIGMOD 2001, New York, NY: 115-126.
[15]. Altinel, M. and Franklin, M. J. 2000. Efficient Filtering of XML Documents for Selective Dissemination of Information. VLDB2000: 53-64.
[16]. Diao, Y., Altinel, M., Franklin, M. J., Zhang, H., and Fischer, P. 2003. Path sharing and predicate evaluation for high-performance XML filtering. ACM Trans. Database Syst. 28, 4 (Dec. 2003):
[17]. Guoli li, S Hou, H A Jacobsen. 2008. Routing of XML and XPath queries in data dissemination networks, ICDCS 2008. Beijing, China: 627-638.
[18]. Petrovic, M., Liu, H., and Jacobsen, H. 2005. G-ToPSS: fast filtering of graph-based metadata. WWW 2005. Chiba, Japan: 539-547.
[19]. E. Fidler, H. A. Jacobsen, G. Li, and S. Mankovski. 2005. The PADRES Distributed Publish/Subscribe System. FIW 2005. Leicester, UK: 12 30.
[20]. P. R. Pietzuch, B. Shand, and J. Bacon. 2004. Composite event detection as a generic middleware extension. IEEE Network Magazine, Special Issue on Middleware Technologies for Future Communication Networks, January/February 2004: 44-55.
[21]. Guoli Li and Hans-Arno Jacobson. Composite subscriptions in content based publish/subscribe systems. 2005. In Proceedings of the Sixth International Conference on Middleware. Grenoble, France: 249-269
[22]. R. Baldoni, C. Marchetti, A. Virgillito, and R. Vitenberg. 2005. Content-Based Publish-Subscribe over Structured Overlay Networks. ICDCS 2005. Columbus, Ohio, USA: 437 446.
[23]. A. Carzaniga, D. S. Rosenblum, and A. L. Wolf. 2001. Design and evaluation of a wide-area event notification service. ACM ToCS. 19 (3): 332-

383.
[24]. Guoli Li, Vinod Muthusamy, and Hans-Arno Jacobsen. 2008. Adaptive Content-Based Routing in General Overlay Topologies, Middleware 2008. Leuven, Belgium: 1-21.
[25]. S. Tarkoma and J. Kangasharju. 2005. Filter Merging for Efficient Information Dissemination. Springer Volume 3760 of Lecture Notes in Computer Science: 274 291.
[26]. A. Riabov, Z. Liu, J. L. Wolf, P. S. Yu, and L. Zhang. 2002. Clustering algorithms for content-based publication-subscription systems. ICDCS 2002. Vienna, Austria: 133-142
[27]. Y.-M. Wang, L. Qiu, D. Achlioptas, G. Das, P. Larson, and H. J.Wang. 2002. Subscription partitioning and routing in content-based publish/subscribe systems. In Proceedings of the 16th International Symposium on Distributed Computing (DISC), Toulouse, France
[28]. Eugene Wu, Yanlei Diao, and Shariq Rizvi. 2006. High-performance complex event processing over streams. In SIGMOD2006. Chicago, Illinois, USA:407-418
[29]. Jagrati Agrawal, Yanlei Diao, Daniel Gyllstrom, and Neil Immerman. 2008. Efficient pattern matching over event streams. In SIGMOD 2008. Vancouver, BC, Canada: 147-160
[30]. Fusheng Wang, Shaorong Liu, Peiya Liu and Yijian Bai. 2006. Bridging physical and virtual worlds: complex event processing for RFID data streams. Lecture Notes in Computer Science, Volume 3896/2006: 588-607.
[31]. Nicholas Poul Schultz-Moller, Matteo Migliavacca, and Peter Pietzuch. 2009. Distributed complex event processing with query rewriting. In DEBS 2009. Nashville, Tennessee, USA. 4:1-4:12.
[32]. BEA Suvey, 2007. Event Processing Market Pulse 2007, On the 3rd EPTS (Event Processing Technology Society) 17-19 Sept. 2007
[33]. http://www.research.ibm.com/distributedmessaging/gryphon.html
[34]. http://padres.msrg.utoronto.ca

**Author Biography:**

Hu Songlin: Associate Researcher, Forward-looking Research Laboratory, Institute of Computing Technology, Chinese Academy of Sciences. husonglin@ict.ac.cn

*Note: Figure translations are in progress. See original paper for figures.*