

ATINER's Conference Paper Proceedings Series

MED2019-0168

Athens, 25 November 2019

Next Generation Blockchain Communication Network (BCN)

Zhibin Lei and Yang Liu

Athens Institute for Education and Research

8 Valaoritou Street, Kolonaki, 10683 Athens, Greece

ATINER's conference paper proceedings series are circulated to promote dialogue among academic scholars. All papers of this series have been blind reviewed and accepted for presentation at one of ATINER's annual conferences according to its acceptance policies (<http://www.atiner.gr/acceptance>).

© All rights reserved by authors.

ATINER's Conference Paper Proceedings Series

MED2019-0168

Athens, 25 November 2019

ISSN: 2529-167X

Zhibin Lei, Director, Hong Kong Applied Science and Technology Research
Institute, Hong Kong

Yang Liu, Hong Kong Applied Science and Technology Research Institute,
Hong Kong

Next Generation Blockchain Communication Network (BCN)

ABSTRACT

The increasing Pico/Femto/Wifi base stations in future generation communication networks are becoming the starting points of mesh P2P blockchain nodes. Three trends are happening:

- 1) Telecom BS are getting smaller in size and covering smaller region with much higher throughput.
- 2) Storage and IDC are being pushed towards the network edge where big data are originally generated.
- 3) Computation resource is getting more parallel and distributed, thanks to the blockchain token mining incentives.

A disruptive revolution is forming for the blockchain based communication network framework with end-to-end P2P based architecture of distributed computation, storage, and networking paradigm – the Next Generation Blockchain Network (NGBN). The core theme of NGBN is to push up communication PHY layer and push down application layer (e.g. storage and computation) to converge on a single networking layer – called Blockchain Network Layer (BNL) – such that blockchain token economy can be efficiently implemented to support an end-to-end P2P mesh network with unlimitedly scalable, available to everyone, and expandable by peer nodes' joining openly, robust and secure network environment for all the IoT, big data, AI applications to be coming in the next 10 years. To achieve this goal, a joint effort is needed to pull together various resources, including network resources, communication resources, application and chip level support, software and system, and initial PoC trials for the end-to-end deployment.

Keywords: Blockchain, Communication, CEP, NDN.

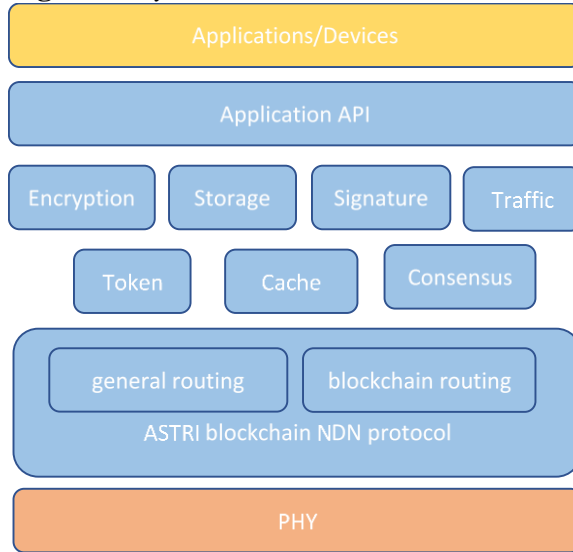
Introduction

The Internet is a global network of a large variety of computer networks. These networks and computers are connected by communications lines and they communicate by using a common transmission language - the Transmission Control Protocol/Internet Protocol (TCP/IP) [1]. Although the Internet carrying a vast range of information resources and services is almost an infrastructure of daily society, the client-server communication model brings some critical issues. For example, services are hard to be deployed dynamically all over the world because they are linked closely with the IP addresses. In addition, the server is an obvious target for hacker attack and the service therefore is easy to fail. Furthermore, a huge inefficiency and a large maintenance cost exists due to the unbalanced network structure [2] [8] [19]. Finally, the cost to process the complex event and data in the system is pretty high due to the adding of huge increase of mobile smart devices and IoT terminals.

On the other hand, since 2015, blockchain rose rapidly [3]. Blockchain is a distributed ledger that is used to maintain a continuously growing list of records, called blocks. Each block contains a timestamp and a link to the previous block. In addition, the information on the chain is encrypted by asymmetric cryptographic methods such as keys and hashing algorithms. By its nature, blockchains are inherently resistant to modification of data once they are recorded. The data stored in any given block cannot be altered retroactively without the alteration of all subsequent blocks and a collusion of the network majority nodes. That is, every blockchain system should resolve consensus problem which means all the participator should assent to the order of blocks and have the same capacity of knowledge about the blockchain of the system. There are many consensus schemes such as PoW, Proof-of-Stake (PoS), PBFT and RPCA and the consensus scheme features the blockchain network [12] [29]. Accordingly, people can communicate with each other easily without any introduction of a third party for endorsement. However, as indicated before, the weak-connectivity and improper protocols cause the propagation delay in IP network. The delay, consequently, causes blockchain forks in some systems. Therefore, in a blockchain communication network (BCN), a novel addressing method similar to named data network (NDN) should be considered. In details, in NDN, desired data is requested by directly using application-defined names, and accordingly named data packets are fetched at network layer [4] [6]-[8] [14] [20]-[22] [26]-[28].

In short, there is an urge need to find a new internet communication structure that performs efficiently and economically to handle new devices and new techniques. In this paper the solution is named Next Generation Blockchain Network (NGBN).

The system framework overview is shown in Figure 1. The blue part is the work scope of NGBN. In the view of protocol, the blue part can be seen as a Blockchain Network Layer (BNL).

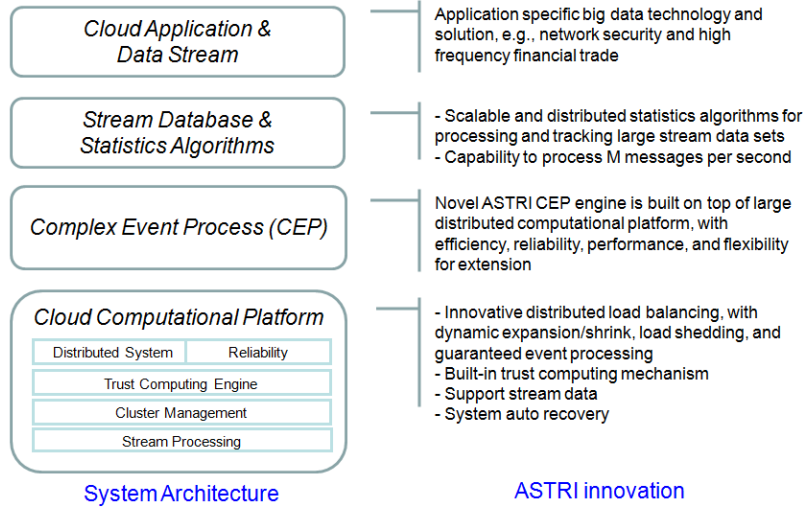
Figure 1. System Framework Overview


Above the BNL, a complex event processing (CEP) platform is designed with a stream data processing engine based on distributed networked machines, which particularly targets to support development, simulation and execution of multiple quantitative strategies in parallel with multiple data or event streams to achieve the goals of four Vs: volume, variety, velocity and veracity, and four Ms: multiple sources, multiple format, multiple depth, and multiple speed. In addition, ASTRI distributed stream computing platform is a general-purpose, distributed, scalable, fault-tolerant, pluggable platform that allows programmers to easily develop applications for processing continuous unbounded streams of data. It accelerates the analysis speed, increases the analysis capacity, and reduces the analysis cost. The previous work can be found in [5] [10] [11] [13] [18] [23]-[25]. The innovation of ASTRI CEP and the distributed stream computing can be seen in Figure 2.

In the BNL, NDN protocol defines how the network works. It specifies the data format and details showing how requests and responses transfer in the network. In details, the routing layer defines the data transfer strategy and algorithm which could keep appropriate transfer flow and efficient data cache mechanism with high level security guarantee. Accordingly, the protocol includes encryption, storage, traffic balance, token control, consensus etc. to ensure the data flow safely transfer with low latency (even for an IoT system under Lora).

The following of this paper can be divided into six parts. In section 2, the smart city application scenario studied for the BCN. Accordingly, a distributed stream computing platform for the applications is illustrated in section 3. Following that, the design to a computational engine for complex event computation and a stream database are explained separately in section 4 and section 5. In addition, section 6 presents the NDN data flow in BNL. Finally, the summary is done in section 7.

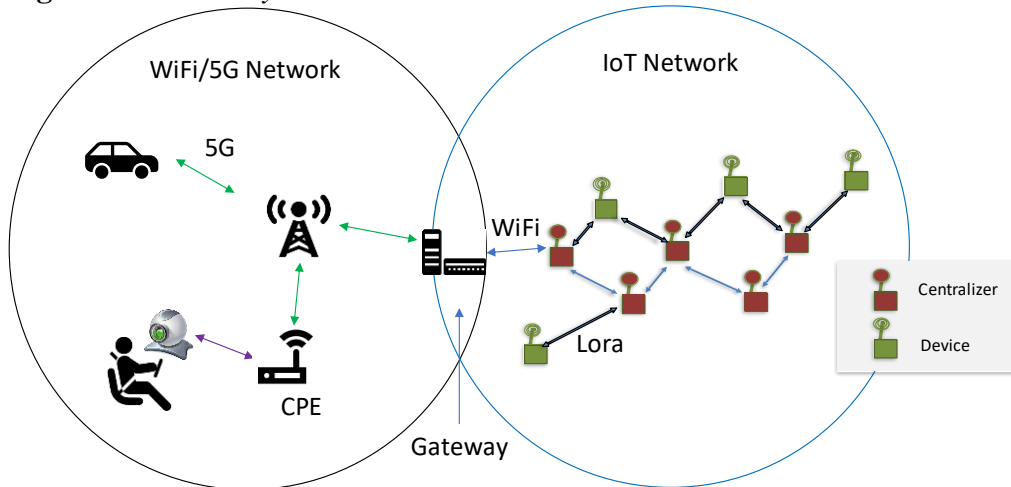
Figure 2. CEP System Architecture



Application Scenario

The application scenario is smart city. A gateway connects the IoT network (e.g. LoraWAN) and the wireless communication network (e.g. 3G/4G/5G/WiFi network).

Figure 3. Smart City Overview



Seen in Figure 3, in the wireless communication side, the vehicles can communicate with the 5G base station and video/audio data captured inside vehicles are transferred to the base station as well. In addition, the IoT devices in this scenario are the sensors installed on the lamp posts collecting various data.

The next generation blockchain is used as the basis of storage and networking in the platform in the middle layer. Above the middle layer, various applications (intelligent surveillance, smart transport, public safety...) can be added. On the

other hand, different devices (vehicles, smart lamp post...) can be introduced to support the applications.

In details, there are three typical cases can be further studied in the scenario:

- **In Vehicle**
The behavior of driving and the interaction between driver and passenger are the study topics. The abnormal items in the topics include driver is sleeping, driver is talking on the phone, and the passenger is quarrelling/ fighting with the driver. With the in-car data collector including cameras, data processor and 5G customer provided equipment (CPE), videos of the driver can be transferred to the server for storage on blockchain or analyzed locally in real time. If there is any abnormal item is detected, the system will release an alert in the vehicle. Furthermore, the driver's mood can be analyzed by the data collected. If the system predicted that the driver is not in a good mood, a reminding may be done in the vehicle.
- **Vehicles on the road**
For the vehicles on the road, the current V2X Vehicle-to-everything (V2X) communication is the passing of information from a vehicle to any entity that may affect the vehicle, and vice versa. There are two types of V2X communication technology depending on the underlying technology being used: WLAN-based, and cellular-based. Since 90 percent of fatal car accidents are caused by human error, assisted driving and connected cars brought by V2X can reduce accidents significantly. That is, V2X paves the way for connected cars and future fully automated driving. And, connectivity with other vehicles, centralized cloud-based back-office systems and roadside infrastructure will enable applications such as hazard warnings and cooperative, adaptive cruise control.
- **Road condition**
The cameras on the smart lamp post can capture the road traffic and analyze the status. All the data captured and calculated is stored in the blockchain network. Additionally, the "helping" action such as sharing of bandwidth for other lamp post's transmission and relay for other node's data can be rewarded with the virtual coins generated in blockchain. Finally, the result found from the current case can be used together with the findings from the two other cases. For example, if the "in vehicle" case releases an alert due to abnormal driving behavior in one bus, the vehicles near the bus in V2X case should receive the alert. If the smart lamp post determines the vehicles receiving the alert is not slow down, the system will send the alert again.

Distributed Stream Computing Platform

Streaming data is all around us – from social media feeds in the cyberspace to the data records in the smart city – but can be difficult to leverage. Sometimes there is simply too much data to collect and store before analyzing it. Sometimes

it's an issue of timing – by the time they store data on disk, analyze it, and respond – it's too late. Organizations need a way to harness the natural resources of streaming data and turn it into actionable insight. In order to make quicker and better decisions, it is not advisable to store the sampled stream data into database then process or analyze. We have to process and analyze such data streams while they just generated or during transmission. Therefore, the major problem that ASTRI computational engine and platform aims at solving is not Big Data Base Management (BDBM), but Big Data Stream Management (BDSM).

There is a lot of emerging applications in which data, generated in a distributed environment, is pushed asynchronously to servers for processing. Some example applications for which this "push" model for data processing is appropriate include financial services (e.g., price feeds), network management (e.g., intrusion detection), medical applications (e.g., monitoring devices and sensors attached to patients) and environmental sensor/actuator systems (e.g., climate, traffic, building, bridge monitoring). These applications are characterized by the need to process high-volume data streams in a timely and responsive fashion. Hereafter, we refer to such applications as distributed stream processing.

What differentiates the distributed stream processing from other big-data solutions is the paradigm that it addresses. Hadoop is fundamentally a batch processing system. Data is introduced into the Hadoop file system (HDFS) and distributed across nodes for processing. When the processing is complete, the resulting data is returned to HDFS for use by the originator. The stream processing supports the construction of topologies that transform unbounded streams of data. Those transformations, unlike Hadoop jobs, never stop, instead continuing to process data as it arrives.

ASTRI has already implemented a distributed stream processing engine that connects to HADOOP to support resource allocation, management and scheduling. Our system is a general platform that can be used in various industries above-mentioned.

User Interface

- End-User Interface
 - Event Composer

This is a graphic user interface for end-users to select input data sources, their format and elements concerned.
 - Rule Designer

End-users can develop their own data processing strategies using pre-defined EPL commands. Rule designer provides the graphical interface to show the list of data elements, EPL commands and elementary strategy functions that end-users can use as building blocks.
 - Case Feedback

In case that end-users use the case-based recommendation system for strategy development, the case feedback system collects the satisfactory data (or success or failure result) of strategy execution.
 - Back Testing

It evaluates the performance of strategies, which end-users created, with the historical testing data provided by the Simulator of the data processing core block.

- Real-Time Dash Board
End-users can monitor the real-time performance of their strategies.
- System administrator interface
 - Event tracker
System administrators monitor the statistics of event processing for all and individual machine in the distributed environment to manage workload.
 - Service monitor
System administrators can start and stop all services and modules. They also monitor the status of each module in the overall platform.
 - Resource monitor
System administrators monitor the status of physical resources in the system cluster.

Complex Event Processing Engine

Complex Event Processing is primarily an event processing concept that deals with the task of processing multiple events with the goal of identifying the meaningful events within the event cloud. CEP employs techniques such as detection of complex patterns of many events, event correlation and abstraction, event hierarchies, and relationships between events such as causality, membership, and timing, and event-driven processes.

Complex Event Processing is an emerging technology for building and managing information systems including:

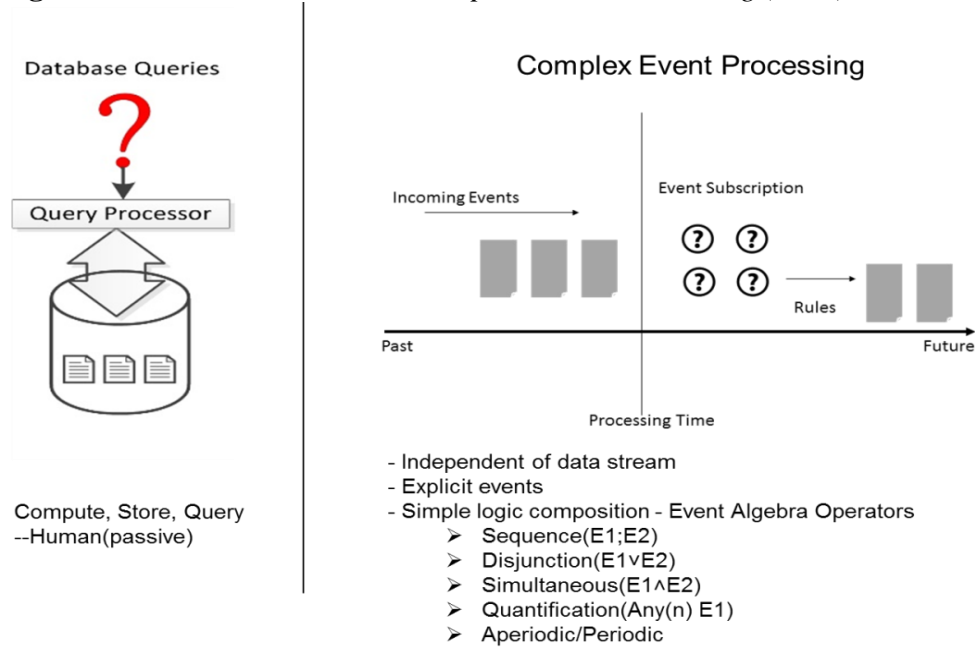
- Business Activity Monitoring
- Business Process Management
- Enterprise Application Integration
- Event-Driven Architectures
- Network and Business Level Security
- Real Time Conformance to Regulations and Policies

Framework Structure

In order to meet the data processing requirements, we combine a CEP engine, which analyzes the correlation among events, with a stream computing system, which enables the scalability in the blockchain network. This system encompasses a real-time data processing engine as well as a batch analysis on historical data to support event correlation and prediction. Relational database usually stores data into a data storage and builds indexing structure to support queries efficiently. However, it cannot address the real time data stream properly because of the large amount of data storage required as well as the latency for building the relational database structure. Stream database management is the new data stream

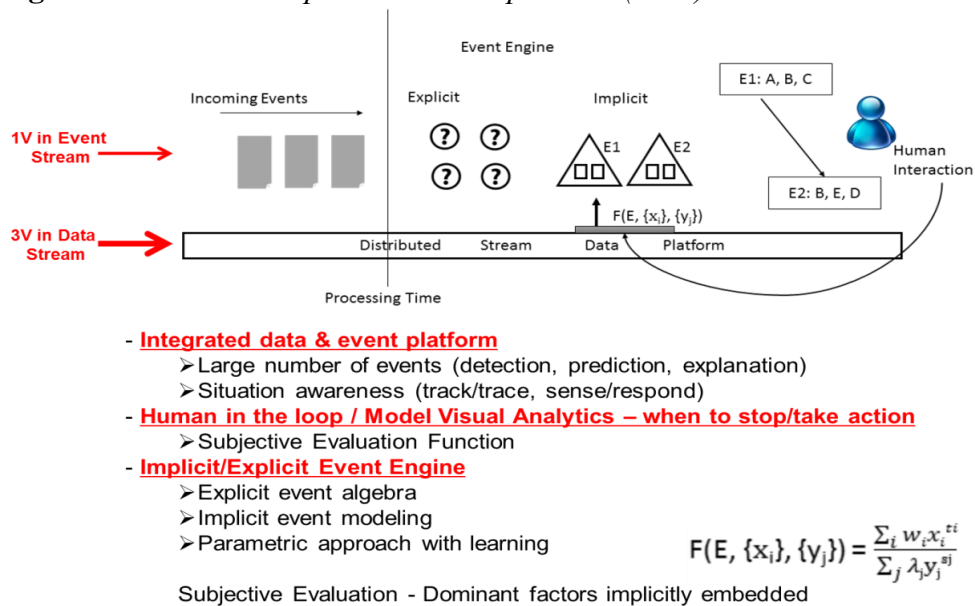
processing platform. It can deal with real time data stream with low latency and fast response.

Figure 4. Relational DB versus Complex Event Processing (CEP)



Event modeling and processing technologies can detect complex events from multiple data streams to generate unordered or unstructured event clouds. Complex Event Computation can produce ordered event streams based on event patterns or rules (explicit or implicit models) for continuous event management.

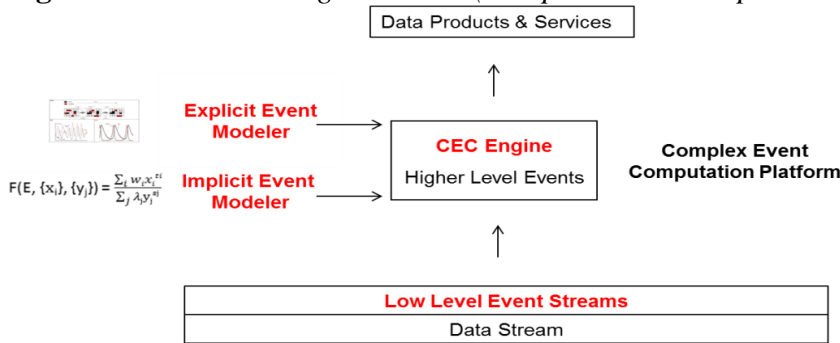
Figure 5. ASTRI's Complex Event Computation (CEC)



ASTRI's Complex Event Computation (CEC) engine has several unique features that are much more powerful than normal CEP engines:

- It is an integrated data stream and event processing platform which can process real time large amount of data and event stream on a united platform.
- It can support interactive function to take inputs/feedbacks from users into the system. For example, it can incorporate subjective evaluation function to model human perception results.
- It can support both explicit event modeling and implicit event modeling. Although explicit model is faster and cost effective, in lots of real world applications, explicit form is not readily available or because uncertainty in data is dominant. Under these situations, implicit model can be used to provide tools and mechanism for event stream handling.

Figure 6. Core Technologies in CEC (Complex Event Computation)



Functional Blocks

Figure 7. Functional Blocks

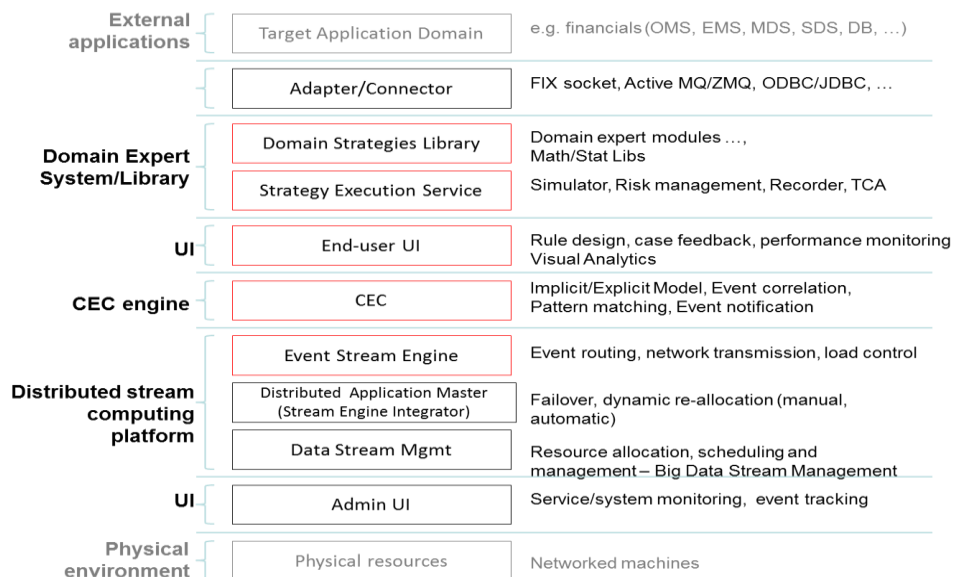
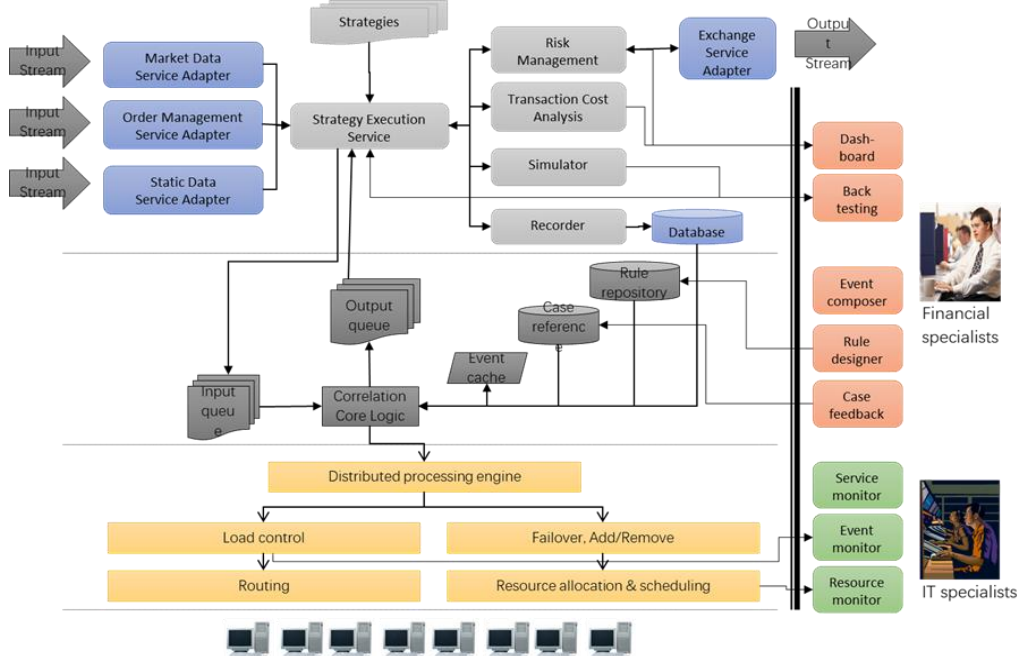


Figure 7 shows the functional blocks of ASTRI CEC system and platform for uncertain data. It is consisted of six functional blocks including two user interfaces: application supporting modules, domain library core engine, CEP engine, end-user UI, system administrator UI and distributed stream computing platform.

Figure 8 shows an implementation example of the functional blocks. The example is a quantitative trading financial application.

- Blue components are the quantitative financial algorithm trading supporting blocks that connect to external stream data sources or database.
- Green components are the quantitative financial core blocks that contain algorithmic strategies library and executors (Strategy Execution Service, Risk management), tester (simulator), evaluator (Transaction Cost Analysis), and recorder.
- Gray components are the CEP blocks that are responsible for event correlation and pattern detection. Purple components are the distributed stream computing platform block that enables the CEP computation to be processed in real-time and manages cluster resources.
- Red components are the end-user interface for financial specialists to compose and test their own strategies.
- Orange components are the system administrator user interface for IT specialists to monitor the status of service components, stream event flow, and physical resource.

Figure 8. Functional Block Implementation Example



Complex Event Processing (CEP) Modules

Event Processing (EP) is a new approach of tracking, processing, and analyzing streams of information, data, and/or even textual information about a specific domain in order to detect or confirm events or things that are happening. Based on the identified and confirmed events, certain decision or actions can be taken. Complex event processing, or CEP, is event processing that with higher degree of complexity of data or information or depth of data or information processing, for example, multiple data or information sources, multiple format of data, different levels of data or information analyzing, etc., to infer complex events, such as terrorist attack or spread of influenza. The goal of using such CEP is to detect useful or meaningful patterns or events and make decision or take action in the earliest possible.

- **Event Processing Language (EPL)**
EPL provides the format to define the processing rules of data for event detection. The rule composed of conditions and actions with temporal constraints. Unlike SQL, it is capable to address unbounded sequencing data (data stream).
- **Event Correlation Engine**
It receives data from multiple data streams and generates output event streams based on the pre-defined event scenarios (in our case, investment strategies). The pattern matcher and temporal sequencer inside the event correlation engine take appropriate actions if conditions are met.
There are various approaches such as FSM (Finite-State Machine), RBR (Rule-based Reasoning), CBR (Case-based Reasoning), MBR (Model-based Reasoning), codebook, voting, explicit fault localization, dependency graphs, Bayesian networks, and ANN (Artificial Neural Network) to implement an event correlation engine. The choice of technology depends on the purpose and properties of the target application. We briefly introduce two most relevant models for our project:
- **Rule-Based Event Correlation**
Users set rules to detect events and/or event patterns that they'd like to take further actions. The rule is mainly composed of condition and action parts. Additionally users can add contexts like priority and scope. Rule-defining languages are in most cases close to natural language, thus the meaning is straight-forward to users and interpretation of events and decision making is clear and transparent. Rules can be created and maintained separately from the engine's source codes, thus changes does not need the compilation of the whole source codes. However, it relies on expert knowledge to define and maintain rules and needs huge efforts to input rules at the initial set-up. Other drawback is that it cannot use past experience for future references.
- **Case-Based Reasoning**
It stores success event-solution (in other words, pattern – action or symptom - suggestion) cases from past experience; therefore enables automatic learning from experience. However, comparing two patterns to

determine their similarity is not simple. If two cases are not perfectly matched but have a certain degree of similarity, solution adaptation to suggest new solution for new event case is tricky. Moreover, it needs a feedback system to the event correlation engine about the success/failure of suggested solution for the event.

Accordingly, we propose a hybrid approach to develop the CEC platform for quantitative finance. As our target applications involve huge investment risks when the system fails, the decision making process of the system must be clear, straight-forward, easy to understand, and reproducible. Therefore, the rule-based CEP engine is most appropriate. On the other hand, a recommendation system that employs the case-based reasoning can exploit the advantage of abundant data resources available in the system. Under this hybrid architecture, we firstly match events based on pre-defined rules. When unknown patterns are detected, we recommend users possible actions (strategies) to take based on case references. The final decision to take those recommendations is up to users.

Stream Database

A real-time database is a database system which uses real-time processing to handle workloads whose state is constantly changing. This differs from traditional databases containing persistent data, mostly unaffected by time. For example, a stock market changes very rapidly and is dynamic. The graphs of the different markets appear to be very unstable and yet a database has to keep track of current values for all of the markets of the New York Stock Exchange. Real-time processing means that a transaction is processed fast enough for the result to come back and be acted on right away. Real-time databases are useful for accounting, banking, law, medical records, multi-media, process control, reservation systems, and scientific data analysis.

In details, stream database are traditional databases that use an extension to give the additional power to yield reliable responses. They use timing constraints that represent a certain range of values for which the data are valid. This range is called temporal validity. A conventional database cannot work under these circumstances because the inconsistencies between the real world objects and the data that represents them are too severe for simple modifications. An effective system needs to be able to handle time-sensitive queries, return only temporally valid data, and support priority scheduling. To enter the data in the records, often a sensor or an input device monitors the state of the physical system and updates the database with new information to reflect the physical system more accurately. When designing a real-time database system, one should consider how to represent valid time, how facts are associated with real-time system. Also, consider how to represent attribute values in the database so that process transactions and data consistency have no violations.

When designing a system, it is important to consider what the system should do when deadlines are not met. For example, an air-traffic control system

constantly monitors hundreds of aircraft and makes decisions about incoming flight paths and determines the order in which aircraft should land based on data such as fuel, altitude, and speed. If any of this information is late, the result could be devastating. To address issues of obsolete data, the timestamp can support transactions by providing clear time references.

Algorithms

Streaming algorithms are algorithms for processing data streams in which the input is presented as a sequence of items and can be examined in only a few passes (typically just one). These algorithms have limited memory available to them (much less than the input size) and also limited processing time per item. These constraints may mean that an algorithm produces an approximate answer based on a summary or "sketch" of the data stream in memory.

The six tracking algorithms of single data stream are implemented. Each algorithm is composed of four basic operations: INITIALIZE, UPDATE, MERGE and QUERY.

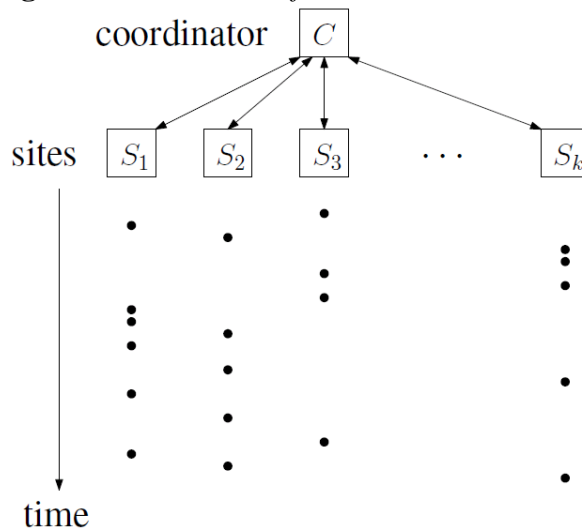
- **Bloom Filter**
Given a candidate item as a query, BloomFilter answers whether the item is in the set or not. It provides a one-sided guarantee: if the item is in the set, then it will always respond positively. However, there is the possibility of false positives, which means it would only make mistakes on items in the set. The probability of a false positive can be bounded as a function of the size of the summary and the size of the set summarized: as the latter increases, false positives become more likely. The summary keeps a bit string, the entries of which are set to one based on a hash function applied to the updates. In its simplest form, only insertions are allowed, but generalizations also allow deletions.
- **MG**
The Misra-Gries (MG) summary retains a subset of items from its input, with an associated weight for each. It answers point queries approximately: the answer to a query x is the weight associated with x in the summary, if x is stored, and 0 otherwise. Given a parameter ϵ , the summary stores $1/\epsilon$ items and counts, and answers point queries with additive error times the total weight of the input.
- **Space-saving**
The Space-Saving summary retains a set of items from its input, with an associated weight for each. It answers point queries approximately: the answer to a query x is the weight associated with x in the summary, if x is stored, and 0 otherwise. Given a parameter ϵ , the summary stores $1/\epsilon$ items and counts, and answers point queries with additive error at most ϵ times the total weight of the input. Conceptually, the SpaceSaving summary is very similar to the MG summary, and in fact the two can be considered almost identical.

- **Count-Min Sketch**
The Count-Min Sketch summary maintains an array of counts, which are updated with each arriving item. It answers point queries approximately by probing the counts associated with the item, and picking the minimum of these.
- **Count Sketch**
The Count Sketch summary maintains an array of counts, which are updated with each arriving item. It answers point queries approximately by probing the counts associated with the item, and taking the median of estimates derived from each.
- **AMS Sketch**
The AMS Sketch summary maintains an array of counts which are updated with each arriving item. It gives an estimate of the Euclidean norm of the vector v corresponding to its input, by computing the norm of each row, and taking the median of all rows.

Distributed Data Stream

The following figure gives a schematic of the model: communication is between the coordinator and the k client sites. Each client sites receive new elements over time, which prompts more communication between the parties.

Figure 9. *A Schematic of the Model*



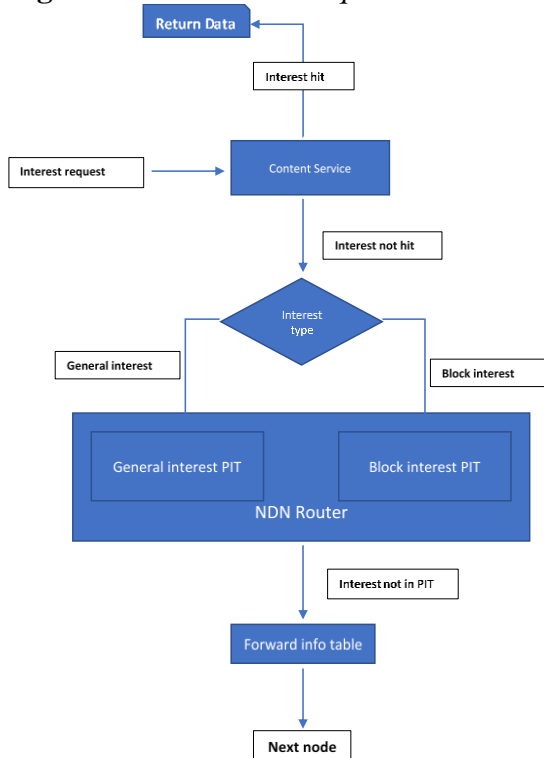
In total we have k distributed client sites, indexed S_1, \dots, S_k , each client site receiving a stream of elements over time, possibly at varying rates. For example, in a communication network, each element might be the arrival of a packet at a router. The description of each event is quite simple: the destination and payload size. But the overall distribution of traffic to different destinations observed over multiple routers is very large and complex. Our goal is to track some of the most important statistics over the union of all the streams. In the distributed data stream model, there is also a server site, called coordinator, C . The coordinator can

communicate directly with each client sites. It has a direct two-way communication channel with each of the client sites.

NDN Data Flow

The request process is shown in Figure 8. The interest packet returns corresponding data directly when the content server hits the matching data. Otherwise, if no matching data is obtained, the interest will be respectively hanged in the corresponding interest list of the NDN router according to the interest packet type. And if it is the first time the interest packet is hanged in the interest list, the interest packet will be forwarded synchronously to the next node. The blockchain interest list and the general interest list should store interests according to different routing policies respectively.

Figure 10. *NGN Data Request Process*



The data from end devices is sent to the NDN router via active pattern or passive pattern respectively (Figure 11). The pattern is determined by the type of data generated specifically. The blockchain data is actively broadcast to the associated router. As a contrast, the general data is pulled to the corresponding router according to the requester's needs. The data transferred to the router will be cached according to the policy in the content server and transmitted to the next corresponding node at the same time. Different data types will have different

caching strategies: blockchain data is stored and broadcast to multiple nodes for long periods of time, while general data is eliminated by cache policy.

Finally, as shown in Figure 12, an enhanced routing device can be implemented. After its generation, the data will be cached directly in the terminal device based on the type. The system will actively push the blockchain data or passively pull the general data to the NDN router according to the request of the node routing respectively. The result is, the system stores the data in the content server and transfer the corresponding data to the next node simultaneously.

Figure 11. *NGN Data Request Process*

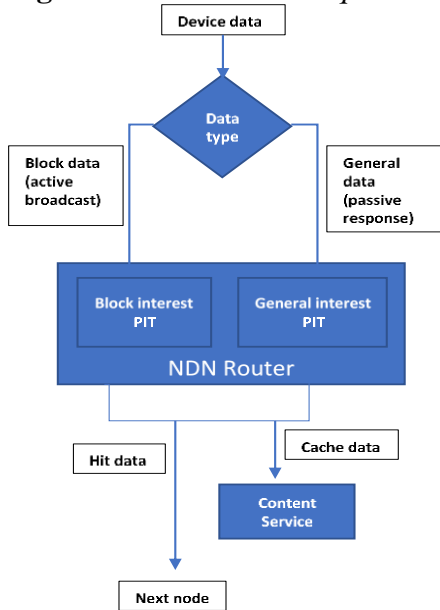
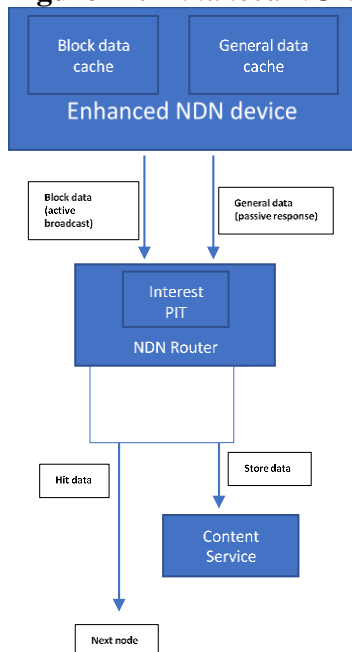


Figure 12. *Enhanced NGN Device*



Conclusions

There are three contributions in this paper. The first one is the CEP platform design with streaming database including the overall structure, the function block and implementation details. The second contribution is the framework design of NGBN. Specifically, the efficient networking based on NDN integrating the IoT application and current wireless communication network is illustrated to direct further industrial trial/implementation. Finally, we studied the smart city scenario under NGBN framework targeting smart transportation.

The future work includes demos on site with various partners.

For example, Hong Kong is a place of small area, high population density and good broadband penetration. In both rural place and urban areas, there are good infrastructures for us to demo NGBN. Therefore, Hong Kong can be a test bed for the new cyberspace.

In addition, in Changsha, Hunan, it is noticed that several 5G base stations are being set up along a road. Accordingly, the wireless connections in NGBN are expected to have a higher rate, a larger density and a lower delay. With the infrastructure improvement, the demo on smart cities will bring the users a better experience in the vehicle related applications.

References

- [1] [Online] *Internet* from Wikipedia. Available: <https://en.wikipedia.org/wiki/Internet>.
- [2] [Online] *PPCoin: peer-to-peer crypto-currency with proof-of-stake*. Available: <http://archive.org/details/PPCoinPaper>.
- [3] [Online] *Executive Summary for Financial Institutions: Ripple Solutions Guide*. Available: http://ripple.com/files/ripple_solutions_guide.pdf.
- [4] Afanasyev, A., Burke, J., Refaei, T., Wang, L., Zhang, B., & Zhang, L. (2018). *A Brief Introduction to Named Data Networking*. MILCOM 2018 - 2018 IEEE Military Communications Conference (MILCOM). <https://doi.org/10.1109/milcom.2018.8599682>.
- [5] Appel, S.; Frischbier, S.; Freudenreich, T.; Buchmann, A.P. Event stream processing units in business processes. In Proceedings of the International Conference on Business Process Management, Beijing, China, 26–30 August 2013; pp. 187-202.
- [6] Bentaleb, A., Begen, A. C., Zimmermann, R., *SDNDASH: Improving QoE of HTTP adaptive streaming using software defined networking*, Proc. ACM Int. Conf. Multimedia, pp. 1296-1305, 2016.
- [7] Burke, J., *Browsing an Augmented Reality with Named Data Networking*, in 2017 26th International Conference on Computer Communication and Networks (ICCCN), July 2017.
- [8] Castro, M., Liskov, B. *Practical Byzantine fault tolerance*, Symposium on Operating Systems Design and Implementation USENIX Association, vol. 20 ,1999, pp. 173-176.
- [9] Dang, H., Sciascia, D., Canini, M., Pedone, F., and Soulé, R., 2015. *NetPaxos: Consensus at Network Speed*. In Proceedings of the 1st ACM SIGCOMM Symposium on Software Defined Networking Research (SOSR '15). ACM, New York, NY, USA, 5:1–5:7. <https://doi.org/10.1145/2774993.2774999>

- [10] Dao, M.; Pongpaichet, S.; Jalali, L.; Kim, K.; Jain, R.; Zettsu, K. *A real-time complex event discovery platform for cyber-physical-social systems*. In Proceedings of the International Conference on Multimedia Retrieval, Glasgow, UK, 1-4 April 2014; p. 201.
- [11] Diallo, O.; Rodrigues, J.J.P.C.; Sene, M. *Real-time data management on wireless sensor networks: A survey*. J. Netw. Comput. Appl. 2012, 35, 1013-1021.
- [12] Donet, J.A.D., C. Pérez-Solà, and J. Herrera-Joanco, *The Bitcoin P2P Network*, The Workshop on Bitcoin Research, vol. 8438, March 2014, pp. 87-102.
- [13] Gyllstrom, D.; Wu, E.; Chae, H.; Diao, Y.; Stahlberg, P.; Anderson, G. *SASE: Complex event processing over streams*. In Proceedings of the Conference on Innovative Data Systems Research, Asilomar, CA, USA, 7-10 January 2007; pp. 407-411.
- [14] Jacobson, V., Smetters, D., Thornton, J., Plass, M., Briggs, N., and Braynard, R., *Networking Named Content*, in Proc. of CoNEXT, 2009.
- [15] Jayasekara, S.; Kannangara, S.; Dahanayakage, T.; Ranawaka, I.; Perera, S.; Nanayakkara, V. *Wihidum: Distributed complex event processing*. J. Parallel Distrib. Comput. 2015, 79-80, 42-51.
- [16] Liu, M. Luo, L., Nelson, J., et. al. 2017. *IncBricks: Toward In-Network Computation with an In-Network Cache*. In Proceedings of the Twenty-Second International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '17).
- [17] Ottenwälder, B.; Koldehofe, B.; Rothermel, K.; Hong, K.; Ramachandran, U. *RECEP: Selection-based reuse for distributed complex event processing*. In Proceedings of the ACM International Conference on Distributed Event-Based Systems, Mumbai, India, 26-29 May 2014; pp. 59-70.
- [18] Perera, C.; Zaslavsky, A.B.; Christen, P.; Georgakopoulos, D. *Context aware computing for the internet of things: A survey*. IEEE Commun. Surv. Tutor. 2014, 16, 414-454.
- [19] Rashid, B.; Rehmani, M.H. *Applications of wireless sensor networks for urban areas: A survey*. J. Netw. Comput. Appl. 2016, 60, 192-219.
- [20] Refaei, T. and Afanasyev, A., *Enabling a Data-Centric Battlefield Through Information Access Gateways*, in IEEE MILCOM 2018.
- [21] Sapio, A., Abdelaziz, I., Aldilaijan, A., Canini, M. and Kalnis, P. 2017. *In-Network Computation is a Dumb Idea Whose Time Has Come*. In Proceedings of the 16th ACM Workshop on Hot Topics in Networks (HotNets-XVI).
- [22] Shang, W., Wang, Z., Afanasyev, A., Burke, J., and Zhang, L., *Breaking out of the cloud: Local trust management and rendezvous in Named Data Networking of Things*, in Proc. of ACM/IEEE IoTDI, 2017.
- [23] Sharples M., Domingue J. 2016. *The Blockchain and Kudos: A Distributed System for Educational Record, Reputation and Reward*. In: Verbert K., Sharples M., Klobučar T. (eds) Adaptive and Adaptable Learning. ECTEL 2016. Lecture Notes in Computer Science, vol 9891. Springer, Cham, Available: https://doi.org/10.1007/978-3-319-45153-4_48
- [24] Stockhammer, T. *Dynamic adaptive streaming over HTTP: Standards and design principles*, Proc. ACM Conf. Multimedia Syst., pp. 133-144, 2011.
- [25] Wu, E.; Diao, Y.; Rizvi, S. *High-performance complex event processing over streams*. In Proceedings of the ACM SIGMOD International Conference on Management of Data, Chicago, IL, USA, 27-29 June 2006; pp. 407-418.

- [26] Zhang, L., Afanasyev, A., Burke, J., Jacobson, V. and et al., *Named data networking*, Newsletter ACM SIGCOMM Computer Communication Review, vol. 44, issue 3, July 2014, pp 66-73.
- [27] Zhang, M., Lehman, V., and Wang, L., *Scalable Name-based Data Synchronization for Named Data Networking*, in Proc. of IEEE INFOCOM, 2017.
- [28] Zhang, H., Wang, Z., Scherb, C., Marxer, C., Burke, J., Zhang, L., and Tschudin, C. *Sharing mhealth data via named data networking*, in Proc. of ACM ICN, 2016.
- [29] Zyskind et. al. 2015. *Decentralizing Privacy: Using Blockchain to Protect Personal Data*, 2015 IEEE Security and Privacy Workshops (SPW), San Jose, CA, USA, July 2015. <https://dx.doi.org/10.1109/SPW.2015.27>