A Survey of P2P Virtual World Infrastructure

Bingqing Shen, Jingzhi Guo and Philip Chen
Department of Computer Science and Information, Faculty of Science and Technology, University of Macau, Macao
{yb17412, jzguo, philipchen}@umac.mo

Abstract—With the development of computer science and virtual reality technology, virtual world evolves along the way of computer game development, from arcade games, console system games, LAN games, Internet connectivity games, unstructured games, games with player generation of content, worlds with designer-provided objectives, games with social networks, and open virtual worlds [1]. Traditional server-client structure does not scale well at least in the following three aspects which are limited number of players in each server, single point of failure risk, and unbalanced computation resource. This survey investigates another alternative, peer-to-peer (P2P) virtual world software infrastructure, to address these traditional architectural issues.

Keywords - Peer-to-peer (P2P), Virtual World, Virtual Environment, Area of Interest (AoI), Scalability

I. INTRODUCTION

With the development of computer science and virtual reality technology, virtual world evolves along the way of computer game development, from arcade games, console system games, LAN games, Internet connectivity games, unstructured games, games with player generation of content, worlds with designer-provided objectives, games with social networks, and open virtual worlds [1]. This evolution pushes the improvement of both computing hardware and software infrastructure, and industry innovation. In the early time, two friends who want to play games together have to look for a common place (either in a game pub or at a house of one of them) to use a single game machine. The birth of Internet changed the situation. With copper or fiber optic cable, two remotely located machines are connected logically. Through LAN, two or more people can share the common screen of the same game world, like StarCraft [2], Counter Strike [3] without walking out of their home. Web 2.0 brought social network to us broadening the relationship among people and making the whole world connected. Group and guild are new social elements which add more diversities to traditional Massive User Dungeon (MUD) games, such as World of Warcraft [4] and Guild Wars [5]. Guild is more like an in-game community where members contribute to the group which conversely will benefit each of them. Open virtual world makes game design more flexible to allow people to customize virtual products and create their own in-world game collectively for fun. The representative of open virtual worlds is Active Worlds [6] and Second Life [7].

Nowadays, millions of people spend significant time on massively multiplayer online role-playing games (MMORPG) or open virtual worlds each day. Due to the huge number of simultaneous online players and uncertainty of player distribution, virtual world infrastructure architecture faces challenges again to address hotspot issue, load balance issue, scalability issue, security issue and the like. Traditional client-server structure [7] does not scale well at least in the following three aspects:

1) **Limited number of players in each server** prevents players’ communication across different servers. Even sharding technique allow more users to play a game simultaneously, but two players in different shards walk in two parallel worlds and cannot meet each other. Hence the scalability is restricted.

2) **Single point of failure risk** is unavoidable in traditional server-client structure. Due to the isolation nature of sharding, if one server is down even due to maintenance or service patch release, players on that server have to wait until the system recovery.

3) **Unbalanced computation resource** restricts the utilization of computation resource. This is a twofold issue. On the server hand, due to the dynamic distribution of players, sharding of servers is not the optimal solution because some servers having fewer players have redundant resources while others having more players facing resource shortage. On the other hand, client machines’ resources are never efficiently utilized.

Since a single machine cannot meet the requirement of contemporary virtual world development. Many researchers turn to peer-to-peer (P2P) [9] architecture for solution. Naturally P2P network structure has the merits in scalability, resource efficient utilization and low cost of deployment, operation, and maintenance. This survey investigates the core P2P virtual world architecture issues and existing methodologies. It starts with the introduction of client-server structured virtual world and the innovation of them to address the scalability issue. The second section discusses the P2P virtual world architecture. It is further divided into three sub-sections, the first part gives some background of P2P technologies and how they gear into P2P virtual world architecture; The second one lists the P2P virtual world platform design requirements as the guidance of problem solving; Then the third part concentrates on some critical challenges in the P2P virtual world architecture. Section 3 introduces hybrid solutions combining both client-server and P2P solution for the complementary of P2P architecture unanswered questions. Finally, section 5 concludes the survey.
II. TRADITIONAL VIRTUAL WORLD STRUCTURE

Traditional distributed virtual world was born from US military simulation system along the history line of DIS [10], SIMNET [10], HLA [11]. In traditional virtual world design, to balance server loads and improve system scalability, different system functions are partitioned into different domains (sometimes called service). In such architecture, one server or a cluster only responsible for one specific type of service and then each kind of service is easy to be scaled simply by adding more servers. Three typical structures are introduced in the following content in this section.

Second Life architecture [12] proposed from Linden Lab is partitioned into agent domain, region domain, and viewer domain. Viewer is a client software handling locations of objects, getting velocities and other physics information, and doing simple physics track of what is moving to where. An Agent Domain [13] handles everything that has to do with agents. It stores user profile data, user inventory, handles login, etc. A Region domain [14] handles everything related to regions, including land information, parcel information, avatar positions, object information (metadata, map information, region statistics), physics, and script execution. The region and agent domain comprise three parts respectively which are service, host and data store. Each part can be hosted by one server or a cluster and linearly extensible.

OpenSimulator [15] is an open-source project providing Second-Life-Style server side software. It separates the service system into five groups [16], which are user service, grid service, asset service, inventory service and message service. User Service is responsible for authenticating the user to the grid. It creates a session identifier for the client which can be used to authenticate requests to the other servers in the same grid. Grid Service maintains avatar or entity geographic location in a region. Asset Service is a data service storing virtual entity data in the virtual environment, including attached sounds, textures, images, notecards, scripts, etc. Inventory Service is another data service responsible for maintaining the hierarchical structure of reference to the entity of asset. Once an asset is created, the inventory server keep track of where it is put. Messaging Service is responsible for establishing text or voice communication in the virtual environment.

OpenWonderland [17], initiate by Sun Microsystems Co. is another framework for building 3D virtual worlds. Open Wonderland framework consists three major modules: Client, Web Server and Service Nodes [18]. Most of the visual logic run on Client – the browser through which user can visually navigate a virtual world. It contains functions of graphic rendering, object collision detection and physics control. The Web Server acts as a central management console, providing web-based management of all services in the system, regardless of what server they are running on. Core Wonderland features operated on the Web server including authentication, asset management, single sign-on (SSO) service. Service Nodes provide world state management and other value-added service such as voice communication, video conference call and other in-world application support.

Service Nodes are highly extendible to enrich a virtual world by adding more sub-modules.

III. PEER-TO-PEER VIRTUAL WORLD ARCHITECTURE

P2P virtual world architecture is still a leading research area. From the very beginning of 20 century, the wave of P2P file sharing and social virtual world application pushes the combination of them to overcome issues in traditional system design. Based on sophisticated P2P underlying communication theories and implementation, virtual world architecting becomes a new branch of the P2P research family. Currently issue of P2P virtual world design mainly focuses on world state (both player incarnation state and object state) management, including state partitioning, state persistence and state consistency, since none of a peer has the global information of the whole world. This section starts introduce the P2P technology itself.

A. Peer-to-peer Technology Background

P2P technology has its wide range of research and application in the last decade, including file distribution (such as Gnutella [19], Napster [20], BitTorrent [21], etc.) and video streaming (such as CoolStreaming [22], P2P Streaming [23], SopCast [24], etc.). Besides, some Web caching systems also designed based on P2P manner [25] to scatter server load to each peer and improve the overall responsiveness. Given that of each peer node in a P2P network has three traits: 1) unfixed role playing – a peer can be a client, server, and router even at the same time; 2) dynamics – a peer is free of joining, leaving the network and changing of its computation capacity and 3) anonymous identity – a peer’s trustworthiness and purpose are unknown, a P2P system design have a strong consideration of tradeoff in three aspects: 1) network structure and topology control, 2) incentive management, 3) trust and security management.

Network structure and topology control research how peers in a network are efficiently organized and the scientific management of peer dynamics, including peer addition, leaving and failure. Prevailing mainstreams of design based on 1) unstructured topology or 2) structured topology. Structured topology works with dynamic hash table (DHT) to match resource identity with peer identity (or node identity), which are centrally assigned and managed. Noted structured topology includes Chord [26], CAN [27], Tapestry [28] and Pastry [29]. Unstructured topology [30] does not have such structure to organize resources. Instead, it relies on flooding query to search a particular resource with the help of a tracker. BitTorrent [21] is the famous example of unstructured topology application. Resource in a structured topology is easier to be located and found due to the avail of global information. But it drops some extent of scalability and autonomy, which becomes the advantages of unstructured topology. Unstructured P2P topology has many other nice characters, e.g. low network diameter and, more importantly, robustness against peer dynamics and random node failures. However, the lack of structure also makes it hard to accurately locate data or peers. Thus many researchers set out to design hybrid network structures to combine the best of both worlds. [31] [32]
For incentive management, altruist and selfishness are a group of contradiction on resource holding and sharing. From a single peer view, it is always trying to grab as much resource as possible from other people which providing as few as possible to save its own computation resource (hard disc, bandwidth, CPU cycle, etc.). Because according to game theory, without knowing other’s information, one always makes a suboptimal decision to protect himself. This is where the incentive management comes from. Since altruist from everyone is just a utopia in peer-to-peer resource sharing scope, a sub-optimal solution is to maintain a healthy and sustainable peer-to-peer network by adopting incentive policies. The main idea is the more resource a peer provides to others, the higher priority it should gain from other peer’s support. Related approaches includes payment-based strategy [33], reciprocity and reputation-based strategy [34][35], penalty-based strategy [36], game theory-based strategy [37], and auction-based strategy [38].

Trust management and security management work on how to establish a trustworthy and secure environment for peer-to-peer transaction by preventing technical attack and fraud. Security and trustworthiness are two vital characteristics in peer-to-peer marketplace building. However, trust means to expose more user information and drop some privacy requirement. Several trust models were established including EigenTrust [39], Peer Trust [40], FuzzyTrust [41], SuperTrust [42], PowerTrust [43] and GossipTrust [44]. Azzedin and Maheshwaran [45] suggested a practical trust computation and management system that is suitable for P2P computing environment and widely cited. In the security domain, P2P systems are naturally vulnerable from distributed denial-of-service (DDoS) attack. Yunhao Liu et al. [46] propose a distributed and scalable scheme, called DD-POLICE to detect malicious nodes to defend P2P system from overlay flooding-based DDoS attack. Brinkmeier et al. [47] proposed several heuristic techniques to make P2P streaming systems more attack resilient from DDoS attack. Other security issues include buffer map cheating [48], Sybil attack [49], and P2P worm propagation [50].

To glue these P2P related technologies, some framework and platform design has been proposed for application developers’ handy use. JXTA [77] prevailing in industries was proposed by Sun Microsystems Inc. is a set of specifications regulate standard API for P2P message transport, peer resource management, group management and security scheme. Bottom up, it has three layers [51]: JXTA core layer, JXTA services layer and JXTA application layer. Core layer provides primitive communication mechanism and group management. Service layer modularize functional components for application utilizing. Figure 1 shows the JXTA layer structure. JXTA does not regulate the topology structure implementation, while P2PS Mozart [52], another peer-to-peer system platform, provides a set of API to separate the implement of underlying network topology and services above it. P2PS Mozart’s algorithms offer strong data lookup guarantees based on structured topology Tango. Through API definition, it can also be adapted to other structured topology such as Chord. Based on the Mozart virtual machine, P2PS Mozart’s structure is similar as JXTA. Other research projects like Chord project [53] and FreePastry [54] are similar as P2PS Mozart, which are just different from programming languages and the network topology structure selection. These platforms all aim at providing a solution to build a scalable, robust distributed application.

B. Requirement of Peer-to-Peer Virtual World Design

Many P2P virtual world architectures contain some requirements of their designers for issues addressing. Doing a comparison among them, J. Chen et al.’s [55] requirement list is relatively complete. It summarized a number of issues involved in designing a networked virtual environment. They are:

1) Consistency: Object states and event orderings received by each participant should be consistent;
2) Responsiveness: The delay of user action and observable consequence should be minimized despite of network latency and heterogeneity of different peer computing capability;
3) Scalability: The number of peers participating the game should be easily added without impacting the network performance;
4) Persistency: Contents within NVEs (e.g., virtual objects and computer characters) may need to exist persistently to provide a sense of realism;
5) Reliability: System should be robust and resistant from hardware or software failures;
6) Security: Account theft and privacy exposure issues should be addressed.

J. Chen et al.’s [55] design based on the VON diagram addressed first three of them. Other designers also raised some requirements in their architectures. T. Hampel’s [56] design focused on scalability, persistency, besides adding the flexibility requirement. D. Ahmed et al [57-61] work on consistency, responsiveness, reliability, persistency and security. B. Knutsson, et al.’s [62] research addressed the responsiveness, reliability and security issue. S. Shen did a thorough investigation in the existing research of P2P game design. He divided the research issue into several categories which can be summarized to: 1) state management, 2) overlay management, 3) content management, and 4) security. The first three issues are related to the system...
requirements of consistency, scalability and persistency. Some of his work will be referenced in this survey.

C. State Management

State management is the critical part of peer-to-peer virtual world design since it directly determines the system’s consistency, scalability, responsiveness, and failover attributes. This section of survey renders the state management in three sub-sections which are state partitioning, state persistency and state consistency. Area of Interest (AoI) management as a supplementary section to the above three aspects will be elaborated at last, because this technique is widely employed to address issues in all the three aspects.

1) State Partitioning

To overcome overload on central server system, researchers and game designers suggest load of object states should be shared by all participating peers. How to effectively divide and manage different portion of world state leads to different design structures. B. Knutsson et al.’s architecture SimMub [62] uses a static partition approach. Geographically neighboring players and objects are grouped into regions. Each region assigned an ID will be run by a host (a peer) and responsible for maintaining the region state among all the regional players. But SimMub architecture did not solve hotspot issue – If many players crowd into one region (let’s say for an in-world event), that host will become overwhelmed with computing resources.

To address the hotspot issue, D. Ahmed et al. presented a dynamic partition model that further divides each region into smaller areas (called microcells) [57][59][60]. And each server (each peer node playing the role of both server and client) is assigned with one or several pieces of microcells. Once a region becomes a hotpot, the management of some microcells will be transferred to other less-crowded hosts for load balance. As shown in figure 2, Server 1 hosts cells (0-0, 1-0, 1-1), Server 2 hosts cells (0-1, 1-2, 2-3) ... and Server n hosts cells (4-1, 3-2, 4-3). When, let’s say, Server 1 is overwhelmed with crowds, and Server 2 has some surplus computing resource, cell (1-1) will be shifted from Server 1 to Server 2.

G. Chandana et al.’s [63] overlapping, zone-based architecture is similar as D. Ahmed et al’s by dividing a region into smaller pieces and adding redundant computing resources to manage each of them. Chandana also proposed a failover solution to address the server crash issue and how resources should be handed over to other hosts when failure occurs.

B. Bossche et al.’s [64] study of dynamic microcell relocations proposes three schemes to optimize microcell distribution and relocation, based on players’ walking itineraries, border crossing rate and load distribution. These schemes should be iteratively executed to reach the final balance state.

Another approach is area-of-interest-based (AoI-based) dynamic partition, which is totally different from the region-based partition. Due to a player’s limited view of the surrounding, he/she only needs to exchange state change information with other players located nearby. Actually, proximity is not accurate because interest area is not only bounded with geographic distance. It will be discussed specifically in the below section.

2) State Persistency

State persistency is important because if there is no such mechanism, when a player shut down his / her machine, all the data running will be lost and that will cause partial game crash. However, due to the best of our knowledge, only a few researches were invested on game state persistency using P2P technique. In M. Kim, et al.’s paper “P2P Second Life: Experimental Validation Using Kad” [66], state persistency holds if no objects gets lost during the evolution of the virtual world. Research by T. Limura et al. [67] worked out a zoned federation P2P model based on the region division approach in which peers are alternately chosen to be the master of a region. The zoned federation model decoupled the game data into transient data (such as player location) and persistent data (such as player’s in-game inventory). The copy of persistent state will be replicated to each peer selected as a zone master. K. Zhang et al.’s [68] design of Mammoth stores persistent data in a relational database. In Mammoth, different objects has different persistency requirement and they are stored differently.

Combining with Mammoth, the zonal federation approach can offer a better persistency solution.

3) State Consistency

Game world state must be consistent among players of a certain range (geographical range or AoI). For example, a bullet fired by player A must be received by the target player B and witnessed by nearby players C and D instantaneously. B. Knustsson et al. [62] divided game state into 1) player state and 2) object state based on different consistency requirement.

Player state update is accessed in a single-writer, multiple-reader pattern. In the paper “A hybrid P2P communication architecture for zonal MMOGs”, D.T. Ahmed et al.’s architecture [57][59][60] combines P2P and multicast techniques to support player state exchange and meantime minimize the range of data transmission. In the hybrid model, each zone has one master node which is the coordinator for intra- and inter-zone communication of that zone. In the operation, all nodes within the same zone register their information to the master node. To lower down
some burden from master nodes, D.T. Ahmed et al.’s model uses dominating set and coloring technique to construct the multicasting mesh among all the non-master nodes. Inter-zone communication occurs when a player approaching the boundary of two zones. To prevent crossing zone messages flooding to all the players in the foreign zone, area of interest is again used to restrict the range of state message broadcasting.

Object state update, however, is more complicated. Kunstsson [62] pointed out because of the collaboration on the same object from multiple players, the final state relies on the result of the common actions. Kunstsson [62] suggested assigning a coordinator in charge of collecting all the information of actions on the object, computing the result and sending it to all the receivers. Then the question becomes how the coordinator system is established free from single point of failure problem and system performance impact.

To prevent plethoric inter-zone communication and keep the continuous view when a player hanging around the border, J. Oliveira et al. [69] suggest an architecture in which part of the foreign zone scene can be embedded to the local zone. In Oliveira’s architecture, foreign zone objects become ghosts in the local zone and their states will be synchronized among all the zones holding this part of scene. Inter-zone communication will only occur when a player really jumps (teleports) to the foreign zone.

D. AOI

Area of Interest is important, especially in large-scale distributed virtual worlds which contains thousands of entities interacting with each other, because it can restrict the range of information exchange, thus resulting in high responsiveness and scalability. D. Ahmed et al.’s publication “Improving gaming experience in zonal MMOG” [60], defined AoI “the visibility scope of player remains inside the zone boundary”. K. Edward et al. [70] interpreted AoI “the radius a user is able to view and directly interact with”. J. Chen et al.’s architecture [55] also employed the AoI concept as a circular-shaped area of which the host player is always located in the center and within which other players should be visible and exchange state information with the host player. In S. Shen’s survey [65], AoI is defined “Due to limitation of human perception, player will only interest with a portion of the game world around his controlled-object. The portion of the game world is called Area of Interest (AoI).”

L. Katherine gave a framework of interest management scheme presented in [71]. In [71], a set of related concepts have been systematically defined. Interest management studies entity’s data interests in terms of location and other application-specific attributes. In interest management, interest expression (IE) is a specification of the data on entity needs to receive from others to maintain state consistency. Entities’ Interest data is transmitted by interest manager (IM) within a specified the domain of responsibility (DoR) or domain of interest (DoI). L. Katherine made observation on seven virtual environment simulation systems which implied 1) interest expressions is always self-referential; 2) interest is not necessarily symmetric; 3) Domain of interest is not always geographically divided though it is the truth in most cases.

However, L. Katherine’s interest management theory is based on client-server structure while peer-to-peer structure double the complexity. A. Yu et. al summarized some research efforts in this area in [72]. These efforts can be classified into two categories in terms of P2P topology: DHT and unstructured P2P approaches. DHT-based approaches utilize network structure to facilitate the dissemination of message by introducing multicast layer or coordinator. In unstructured approaches, entities maintain neighbors’ states more autonomously. Three proposals of unstructured approaches were mainly discussed. They employ number of hops in P2P network or geographical distance in the virtual world to derive peer’s AoI. VON P2P network architecture is the famous one of them. J. Chen [55] and S. Hu et al. [73] give the details of how to apply Voronoi diagram in P2P area of interest management. In J. Chen design, it defines a set of concepts (illustrated in figure 3 (b)) regarding the AoI first, including:

- AoI Neighbor: the nodes whose positions are within its AoI;
- Boundary Neighbor: nodes whose regions immediately surround the given node;
- Enclosing Neighbor: AoI neighbors whose enclosing neighbors may partially lie outside the AoI.

In the publication “VON: A Scalable Peer-to-Peer Network for Virtual Environments” [55], J. Chen, et al. bounded player’s AoI with voronoi diagram to decompose a space. Figure 3 (a) illustrates a space decomposition based on Voronoi diagram. Within boundaries of a region, all the distances to the bold dot are shorter than to the bold dots outside the region. Voronoi diagram spatially optimize the range and shape of AoI within which messages can be efficiently broadcasted to limited listeners. S. Hu et al. in [73] elaborated the procedure of applying Voronoi diagram in a P2P networked environment, discussing the player joining, moving, leaving actions adapted in a Voronoi-based AoI division.

![Figure 3. Voronoi diagram](attachment:voronoi.png)
As aforementioned, L. Katherine pointed out entity interest could be a multi-dimensional domain other than only geographical location [71]. K. Edward, in the design of VELVET [70], used a set of metrics to regulate the scope of AoI. For instance, a set of metrics are selected MS = {M1, M2, ..., Mn}. For example, Metric 1 (M1) can be geographical distance; Metric 2 (M2) can be upload bandwidth, and the like. Objects only meeting the all the requirements will be visible inside the AoI. The image of the world captured in each player’s AoI is called parallel virtual world (PVW). To prevent frequently change of communication list in one player’s AoI due to, for example, objects’ movement or bandwidth change, VELVET regulates an upper bound and lower bound value for each metric. When an object is moving (using geographic distance as the only metric) toward the direction leaving the AoI from inside, it will only move out after it crosses the upper bound. When an object is moving towards the direction entering the AoI from outside, it will go inside only when it crosses the the lower bound. To visualize these two movements, in figure 4, each of them is represented with two rings with different radius. The outer one represents the upper bound and the inner ring is the lower one. In figure 4, an object is moving along the border of AoI. The movement trajectory is highlighted with a squiggly curve, and the crossings of the border indicated with arrows and numbers. It can be observed that with two borders (figure 4 (B)), the number of crossing is reduced comparing with the single border case (figure 4 (A)), and it becomes less when the distance of two borders increases (figure 4 (C)).

**Figure 4. Visualization of VELVET’s AoI boundaries.** [70]

### IV. HYBRID VIRTUAL WORLD STRUCTURE

Due to the lack of central repository of persistent data, there are some critiques on peer-to-peer-based virtual world design. Comparing with traditional server-client structure, at least three drawback of can be concluded from existing peer-to-peer virtual world design: Lack of central administration makes state consistent hard to maintain globally; Lack of central organization increase the difficulty to globally optimize the computation resource arrangement from heterogeneous capacity and bandwidth of peers; High dynamics of peer causes high communication overhead. Therefore, we can found some researchers dedicating to mix two worlds together and build a hybrid virtual world architecture. Based on the traditional server-client structure, K. Kim, et al. [74] introduce cell-daemon-ation concept to reduce the overall server-side bandwidth consumption. In his demo system HYMS, a virtual world is divided into several cells. In each cell, some qualified client nodes manages in-cell state update incorporating with server daemon threads (central cell-daemons, or CCDs). Some of these clients picked to play some of the server roles are called master cell daemon (MCD) and each of them manages one of the cells. And other qualified nodes in the same cell are the backups of master nodes, called secondary cell daemon (SCD).

C. Bezerra et al. proposed a virtual world region partition but on server side instead of client side based on the K-d tree division of virtual world regions. In A. Marios et al.’s design [76], server-side manages microcells dividing the world regions while client side peers compute the AoI and maintain object state in corporate with each grid server. C. Chris et al. [75] expanded existing virtual world framework Homura by adding peer-to-peer layers both on client and server side. Many other massively multiplayer online role-play games (MMORPG) introduce transfer the subsidiary functions such as multimedia chatting data in peer-to-peer manners to lower down server balance.

### V. CONCLUSION

This paper started reviewing the virtual world system development history and issues of existing architectures due to the intrinsic features of server-client technologies. Then it introduced some approaches of existing efforts to overcome some of the issues with three cutting-edge examples. Peer-to-peer system, naturally with strong scalability, as another alternative is mainly discussed and anticipated to be the substitution of client-server architecture in future virtual server design. The critical issue in P2P virtual world system is to maintain world state consistency, persistency and system load balance. They are addressed through world resource proper division and management of each player’s area of interest. The aim is to maximize each peer node’s resource utilization while minimize the communication overhead between them. Some hybrid models are also investigated for reference to solving P2P hard nuts.

Finally, P2P virtual world is different from P2P file sharing and video streaming due to its high state consistency and constantly real-time update requirements. From This survey, we draw some research issue of virtual world infrastructure, which includes but not limited:

- The network topology best fitting the client dynamics requirement in virtual world application, structured topology, unstructured topology or even hybrid one.
- The most efficient strategy for world state dissemination, which should support maximizing the responsiveness while minimizing the communication overheads.
- The best strategy for world content distribution to prevent content lost in case of retreat or failure in single peer.
- Performance metrics to evaluate the architecture and operation, including in AoI, the attributes based on which an entity’s interest data will be filtered.
- The tradeoff between pure P2P structure or hybrid structure which mitigate but not eliminate single point of failure risk at server side.
Security issue in P2P virtual world operation.

ACKNOWLEDGEMENT
This research is partially supported by the University of Macau Research Grant No. MYRG156(Y2-L2)-FST11-GJZ.

REFERENCE


