Accepted Manuscript

Software agents for peer-to-peer video streaming

Kemal Deniz Teket, Muge Sayit, Geylani Kardas

DOI: 10.1049/iet-sen.2013.0181

To appear in: *IET Software*

Published online: 25 February 2014

Please cite this article as: Kemal Deniz Teket, Muge Sayit, Geylani Kardas, Software agents for peer-to-peer video streaming, IET Software, doi: 10.1049/iet-sen.2013.0181

Software Agents for Peer-to-Peer Video Streaming

Kemal Deniz Teket, Muge Sayit, Geylani Kardas

International Computer Institute, Ege University, 35100, Izmir, Turkey

denizkemal@gmail.com, muge.fesci@ege.edu.tr, geylani.kardas@ege.edu.tr

ABSTRACT

Peer-to-peer (P2P) video streaming systems enable video data exchange between peers by reducing the overload of the servers while utilizing network resources. Multi-agent systems (MAS), including many autonomous and intelligent software agents working on behalf of video peers, may also provide a feasible infrastructure for the implementation of P2P video streaming systems. Within this context, research on the use of software agents in video streaming systems has recently emerged. In this paper, we discuss the development of an agent-based P2P video streaming system. Software engineering of the MAS with covering various aspects such as agent internals and interactions in the system is described. Performance evaluation of the proposed agent-based P2P system comparing with the popular in-use streaming application is also reported. Results show that a well-defined behavior of a parent selection software agent can improve the continuity index in P2P video streaming systems. Therefore, the users in the system can watch video in a better quality and lower end-to-end delay comparing with the currently used video streaming applications. We also examine that peer welcoming (traffic localization) behavior of the agents decreases the internal Internet service provider (Inter-ISP) traffic significantly.

Keywords: Software agent, agent-oriented software engineering, peer-to-peer (P2P), video streaming

1. INTRODUCTION

Nowadays, much of the network resources are consumed by video streaming applications running over the Internet. Peer-to-peer (P2P) video streaming systems enable video data exchange between peers by reducing the overload of the servers while utilizing network resources. Hence, it is possible that large number of peers enjoy video streaming. An end-user plays video over the Internet and also acts as a content provider to other end-users by using popular P2P live streaming applications (e.g. Coolstreaming [1], PPLive [2], PPStream [3], UUSee[4], SopCast [5]). Furthermore, time constraints, minimization of the delay and efficient utilization of network resources should be taken into consideration during ideal P2P streaming. We believe that software agents can be employed inside P2P video streaming applications especially during both scheduling video data dissemination and efficient network resource allocation.

Multi-agent systems (MAS) [6], including many autonomous and intelligent software agents working on behalf of video peers, may also provide a feasible infrastructure for the implementation of P2P video streaming systems. Within this context, research on the use of software agents in video streaming systems has been recently emerged [7-11]. However parent selection and network awareness are not addressed in those leading studies. In order to fill this gap, we discuss the development of an agent-based P2P video streaming system in this paper. Software engineering of the MAS with covering various aspects such as agent internals and interactions in the system is described in the paper. Performance evaluation of the proposed agent-based P2P system comparing with the popular in-use streaming application is also reported.

The rest of the paper is organized as follows: Section 2 presents a brief description of the P2P video streaming for readers who are unfamiliar to this domain. Software engineering of the proposed MAS within the scope of the applied methodology is discussed in Section 3. Section 4 includes the evaluation of the proposed system. Section 5 discusses the previous studies which make use of

software agents in P2P video streaming applications. Finally, we conclude and describe future work in Section 6.

2. PEER-TO-PEER VIDEO STREAMING

P2P video streaming applications such as CoolStreaming [1], PPlive [2], PPStream [3], UUSee [4], Sopcast [5] enable users to watch video while collaborating for the dissemination of the video data. In these systems, nodes help to reduce server(s) load since video streaming is based on sending video data between peers.

In this section, we discuss CoolStreaming-like P2P video streaming and explain P2P video streaming challenges. In order to evaluate the performance of the proposed MAS over a comparison with the current classic P2P systems, we chose CoolStreaming as the opponent system model because it has lower message complexity than pure pull systems while providing higher quality of service. Coolstreaming is also one of the most popular and industrious P2P video streaming applications widely used especially in China. In the following, we first list the basic terms and then discuss the operation mechanism of P2P video streaming. Later, challenges of the domain are given in a separate subsection.

Membership (MS): This is also known as the partial view of the overlay. Peers have MS lists. Peers in a MS list do not have to communicate with each other.

Partnership (PS): PS list is selected from the MS list. PS is mutual and partners exchange buffermaps periodically.

Parent: A peer gets video from its parents. Parents are the subset of the PS List and a peer controls its parents periodically in case of peer churn and low bitrate.

Substream: One stream is divided into multiple substreams. Each substream can be downloaded from different parents. Hence, if one stream is decomposed into e.g. eight substreams, a peer can have at most eight parents.

Buffermap (BM): Buffermap represents the availability of the latest blocks of different substreams in a buffer. BM information is exchanged periodically among partners to select/re-select parents and to control status (online/offline). The buffermap vector records the sequence number of the latest received block from each substream.

Heartbeat (HB): Peers have to communicate with a Tracker Server periodically to inform its status (online/offline). This is called Heartbeat Message.

In live P2P video streaming systems, there must be at least one video server and one tracker server. The video server initiates video distribution in the system. The tracker server provides a partial list of the currently active nodes to newly joined nodes. This partial list is called MS list.

A newly joined node communicates with its members in order to establish the PS list. Then it selects a few of them to initiate video download. The partner and the parent are different. A peer does not have to download any video content from one of its partners but they must communicate periodically in order to exchange the information on video availability. If a peer (child) selects one of its partners as a parent, it means the peer is downloading video data from that partner (parent).

In push-pull based mesh streaming, the video data are partitioned into k substreams and each node subscribes to one or more parents to receive these k substreams. In CoolStreaming, the nodes in the system periodically exchange their BMs which show buffer position for each substream between them and their partners. Parents are selected from the set of partners according to their BM values. During streaming, BMs of both partners and parents are compared periodically in order to detect insufficient video retrieval.

The video dissemination model is based on a push-pull mechanism as illustrated in Figure 1. Here, the video is partitioned into 4 substreams. Peer 3 subscribes to Video Server for substream-1, substream-2 and substream-4 (as shown in the list just below the outgoing arrow from Peer 3). Subscription process can be defined as pull mechanism. Video server registers Peer 3 as a child and starts sending video. Additionally, Peer 42 has two children: Peer 13 and Peer 84. Peer 42 sends substream-1, substream-2 and substream-3 to Peer 13 and substream-1 and substream-4 to Peer 84 (push) (see the list at the left of Peer 42). This sending process can be defined as push mechanism. Peer 42 sends video to Peer 13 and Peer 84 until it receives an unsubscribe message.

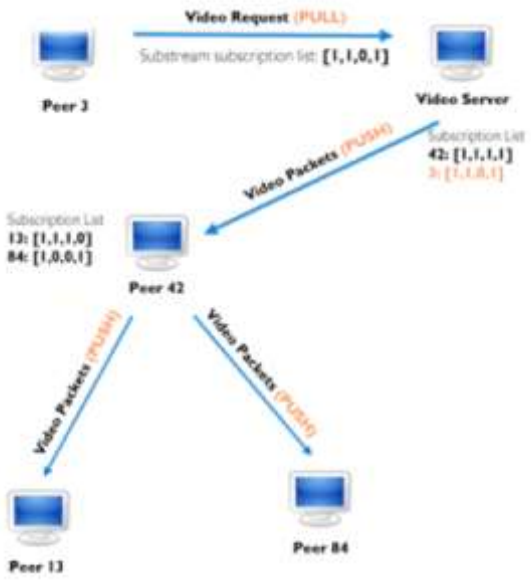

Figure 1. Push-Pull mechanism

Once a peer (child) selects its parent, it sends video request to its parent. Parent initiates the video stream. Until the peer (child) sends unsubscribe message to its parent, parent continues to send video data. Peer also controls its parent's performance periodically since parent may leave the system (peer churn), or the video download rate may decrease due to the network dynamics. If the performance of its parent decreases, peer compares BM information of its partners and selects a new parent among its partners.

This control mechanism provides a node to detect congestion or inadequate bandwidth of a link in the path between the source and itself. Parents, who fail to meet these requirements, are changed with any node from the partners whose BM value for the requested substream is consistent with the node's BM. For instance, in CoolStreaming, if the BM values of more than one node are consistent with the node requiring the parent change, then one of them is selected randomly.

2.1. P2P VIDEO STREAMING CHALLENGES

Similar to the file sharing systems, the main challenge of the P2P systems in general is to provide a system which is both distributed and scalable by considering the upload capacity distribution of the peers and peer churn. The negative effects of upload capacity limitations and the peer churn can be minimized if smart algorithms are implemented in peer selection process.

In addition to abovementioned common challenges for all P2P applications, specifically P2P video streaming systems are not tolerant to packet delays. Packets received later than playout deadline are considered as lost in P2P video streaming systems. Therefore timing constraints must also be considered. Furthermore, startup delay should be minimized. That means a user must not wait too long to start watching video. Also, the interval between video packets arrival time should be similar, in other words, jitter should be as low as possible in order to play video smoothly.

In P2P video streaming systems, peer churn influences system performance much more when compared to ordinary P2P file sharing systems. File sharing has no timing constraints, so peer churn does not affect P2P file sharing systems as much as P2P video streaming systems. Peer churn must be specifically considered in P2P video streaming due to timing constraints. An approach based on autonomous software agents may pave the way for the construction of a software system in order to cope with those challenges. Following sections discuss the design and implementation of such system.

3. SOFTWARE ENGINEERING OF THE PROPOSED AGENT-BASED P2P VIDEO STREAMING SYSTEM

In order to develop the agent-based P2P video streaming system, we apply an agent-oriented software engineering (AOSE) methodology called Prometheus. Prometheus [12] is a well-known detailed process for specifying, designing and implementing MAS. It also defines various artifacts to use during and after the design of the agents. As depicted in Figure 2, Prometheus Methodology has three phases. In the System Specification Phase, goals, scenarios, and basic functionalities are described. Inputs and outputs are defined. In the Architectural Design Phase, agent types and interactions are defined using scenarios and functionalities. Finally, in the Detailed Design Phase, the internals of the agents are elaborated.

During the design process of our system, we employ RMIT Intelligent Agents Group's graphical Prometheus Design Tool (PDT) [13] which supports Prometheus methodology. We also use Network Simulator 3 (ns-3) [14] and C++ programming language to implement the required network overlay and the proposed agent design respectively. Following subsections discuss software engineering of the proposed MAS in detail within the scope of the Prometheus AOSE methodology.

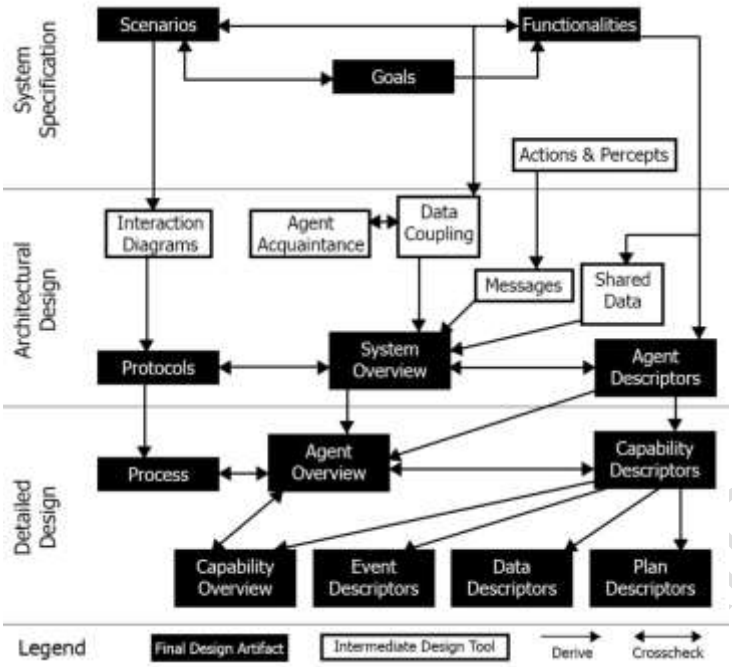

Figure 2. The phases of the Prometheus Methodology (adapted from [12])

3.1. System Specification

Conforming to the Prometheus methodology, we first define system goals, scenarios and functionalities in the System Specification phase. We also describe the interaction between agent system and the environment (percepts, actions, and data sets). In Figure 3, System Goal Diagram is shown. Main agent goals, their related functionalities and scenarios within our system are described as follows:

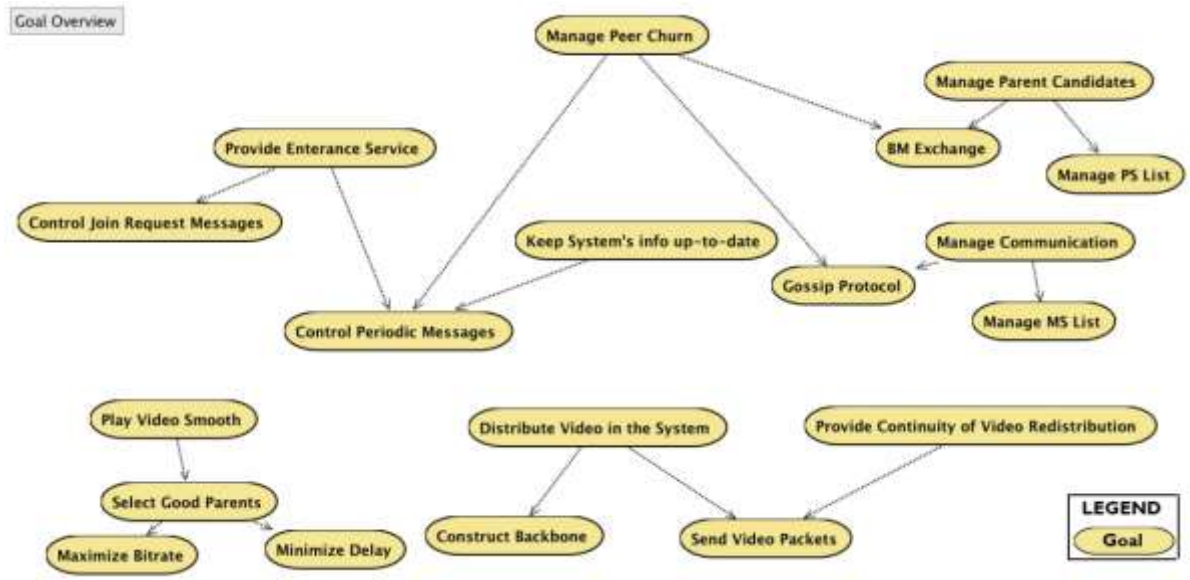

Figure 3. System goal diagram

Provide Entrance Service: The newly joined peer needs to communicate with other peers in order to download video data and it sends join request to the bootstrapping node (tracker) to obtain a list of peers.

Keep System's Information Up-To-Date: In order to keep the system up-to-date, the tracker updates peers' information via heartbeat messages.

Distribute Video in the System: Video data needs to be distributed from a source during the live video streaming. Therefore, there is a video server in the system and it must select good peers having high upload capacity and different Internet Service Providers (ISP) to construct a solid backbone.

Manage Communication: A peer in the system has a MS list which is a partial view of the system, and it communicates with members to build the PS list.

Manage Parent Candidates: A peer chooses parents from the PS list using BM and previous information about partners.

Play Video Smooth: A peer needs to download video data on time to play the video smoothly. Therefore, it must choose appropriate parents to maximize bitrate and minimize delay.

Provide Continuity of Video Redistribution: All peers in the system must buffer video data and send this data to its children.

Manage Peer Churn: All nodes in the system must update their PS and MS lists in case of peer churn.

3.2. Architectural Design

In the Architectural Design phase, we use artifacts (scenarios, functionalities, actions, percepts) from the system specification phase and describe agent types, interactions between agents (by using interaction diagrams) and system's overall structure (by using the system overview diagram).

In our design, we define 3 agent types: Tracker Server, Video Server and Peer agent types and functionalities are coupled as follows: While Tracker Server agent provides Peer Maintenance Service and Peer Information Service, Video Server agent is responsible from the Video Distribution Management. Finally, Peer agent owns the following functionalities: Communication Management, Partnership Management, Video Download Management, Video Redistribution Management and Peer Churn Management. We also define interactions between agents, such as "System Join", "Partnership", "Buffermap Exchange", "Video Request", "Receive Video", "Control Parent", "Change Parent", "Update Peer Lists".

The diagram pertaining to "System Join", "Partnership" and "Video Request" interactions are shown in Figure 4. Peer agent begins with the "System Join" percept. It sends "Membership Request" to Tracker agent. Tracker agent sends "Membership Reply" to Peer agent in response to "Membership Request". In "Membership Reply", there is a list of peers, which are online in the system. After getting Membership List, Peer agent sends a "Partnership Request" to all peers in its Membership List. Partnership is mutual; therefore if the receiver peer's Partnership List is not full, it accepts the request and sends "Partnership Reply" to the newly joined peer. Thus, the newly joined peer builds

its Partnership List. Each Peer agent in the system sends periodically “Buffermap” information to its partners. Using the partners’ BM information, Peer agent selects its parent in order to download video data. It also controls the performance of its parents periodically to find better parents.

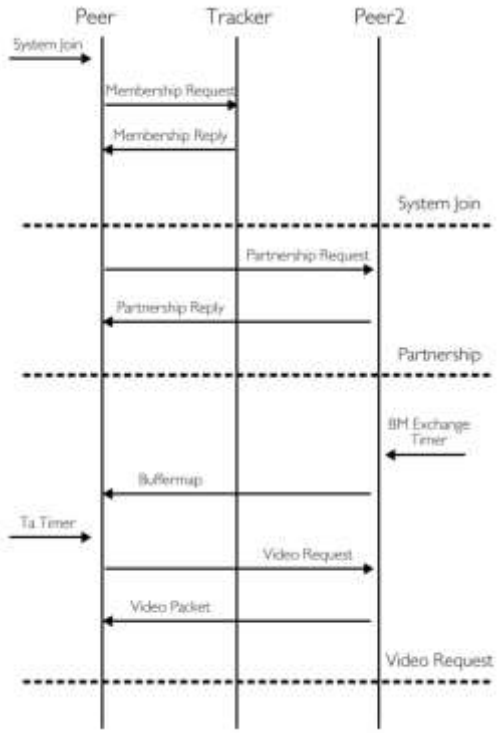

Figure 4. System Join, Partnership, and Video Request interactions between the system agents

The System Overview Diagram for our MAS design is shown in Figure 5. This diagram is perhaps the most important artifact of the design process, since all required agent types, percepts, actions, messages and interactions between agents can be described. For example, in the “Update Peer Lists” scenario, a Peer agent’s “BM Exchange Timer” expires and yet it does not receive “BM Message" from one of its partners. In this case, Peer agent excludes this peer from its “MS List” and “Partnership List” and sends “Gossip Message” to its partners. In “Gossip Message”, there is a list of offline peers, and a list of newly joined peers. The whole system updates itself with “Gossip Message”.

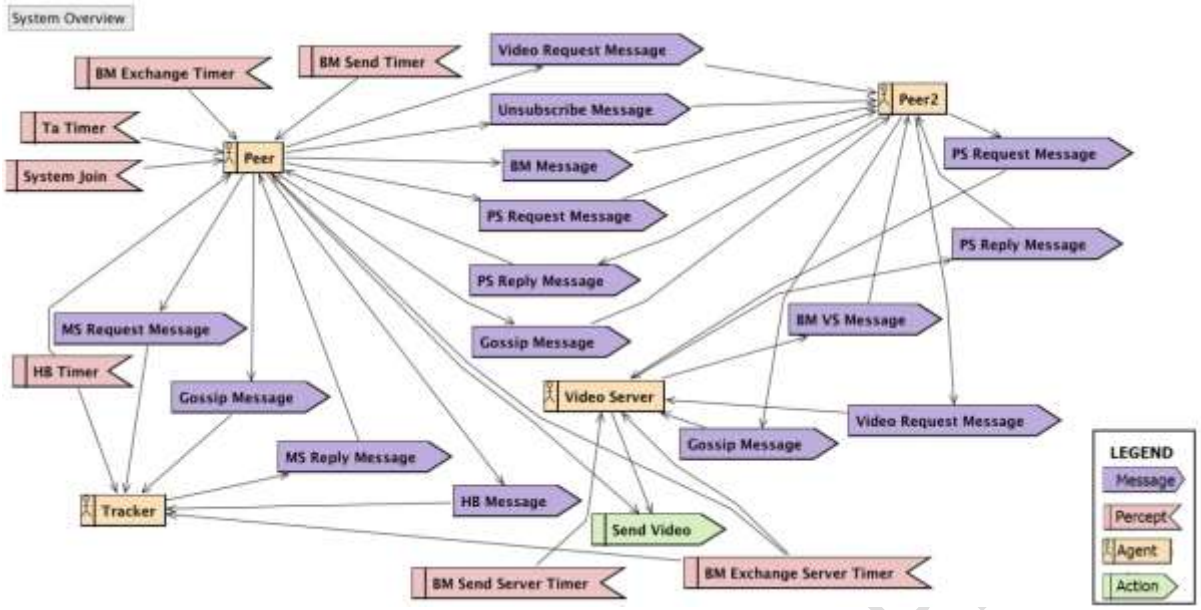

Figure 5. System overview diagram

3.3. Detailed Design

In the Detailed Design Phase, we use artifacts from the previous phases and design internals of the agents. We describe capabilities, plans, events, and data of each agent in the proposed MAS. It is also worth noting that we employ the well-known Belief-Desire-Intention (BDI) architecture [15] in order to both design and implement the internals of our software agents. In a BDI architecture, an agent decides on which goals to achieve and how to achieve them. Beliefs represent the information an agent has about its surroundings within a beliefset, while Desires correspond to the goals that an agent would like to be achieved. Intentions, which are deliberative attitudes of agents, include the agent planning (behavior) mechanism in order to achieve goals. Detailed design of each agent type is discussed individually in the following subsections.

3.3.1. Peer Agent

As illustrated in Figure 6, Peer agent has 5 capabilities as briefly discussed below:

MS Managing: In this capability, Peer agent manages “Membership List”. If peer is newly joined, it builds MS list, or it can update MS list according to location information.

PS Managing: In PS Managing capability, Peer agent manages the “Partnership List”, and related functions. It builds PS list, updates PS list according to the location and upload capacity information. It is also responsible from BM exchange and HB messages.

Churn Handling: Peer agent periodically controls other peers’ online status and updates its lists within this capability. Related update is performed when the agent receives a “Gossip Message” or “BM Exchange Timer” expires.

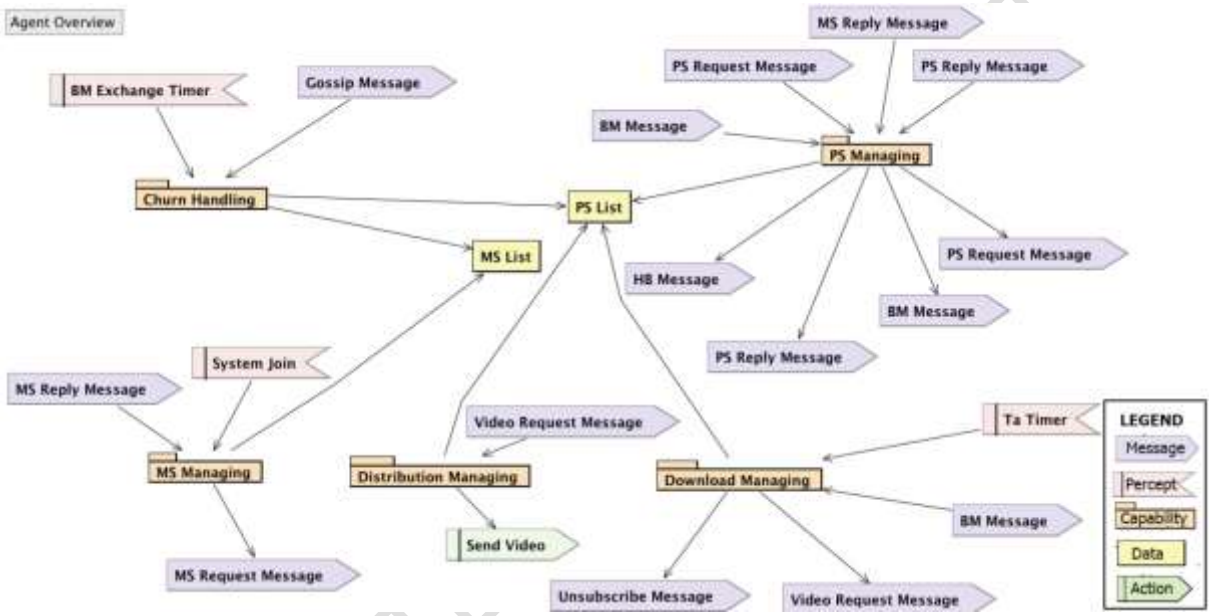

Figure 6. Overview diagram of Peer Agent

Download Managing: In this capability, Peer agent manages its parents and updates partners’ information periodically via “Buffermap”. In addition to the "Buffermap", update of parents’ information can also occur as the result of the evaluation on bitrate and delay values of the parents.

A Peer agent applies a parent scoring method within this capability. During its Parent Scoring behavior, a Peer agent gives points to its parents according to their bitrates and end-to-end delays. Higher bitrate and lower end-to-end delay get higher score. The parent scores are stored cumulatively. For example, if the bitrate is higher than the bitrate threshold and end-to-end delay is lower than the delay threshold, parent gets 4 points. If the bitrate is higher than the bitrate threshold

but end-to-end delay is higher than the delay threshold then the agent gives 3 points to the parent. Detailed discussion on this scoring approach can be found in [16].

Distribution Managing: Peer agent manages video distribution with this capability. It uses the list of members ("MS List") for this purpose. Members are potential partners, but Peer Agent does not have to communicate with members. MS List can also be named as the partial view of the system. The structure of the MS List has two parts: PeerID (a unique ID that is given by Tracker Server) and IPAddr (IP address of the peer).

On the other hand, Peer agent must communicate with its partners periodically to inform about the video availability and the system status (online/offline). That communication is performed by using the list of partners (“PS List”). Partnership is mutual. In the PS List, partners also have rankings. One partner may have higher rank if it is in the same ISP or sends video data on time. The structure of the PS List is composed of PeerID, IP Addr, SameISP, Buffermap, ParentScore, ParentInfo, ChildInfo and ChildBuffermap. PeerID is a unique ID given by Tracker Server and IPAddr is the IP address of peer. SameISP is a flag for ISP information. If it is “true”, the peer and the partner are in the same ISP. Buffermap is an array of four integers and keeps partner’s BM information. ParentScore stores parents’ scores cumulatively using above discussed behavior. The default value is set to zero. ParentInfo is an array of four booleans and keeps parent information of substreams. If it is “true”, the peer downloads the substream from the partner. ChildInfo is an array of four booleans and keeps child information of substreams. If it is “true”, the peer sends the substream to the partner. Finally, ChildBuffermap is an array of four integers and keeps the information of packets to be sent.

Peer agent tries to achieve 5 goals: Manage Communication, Manage Parent Candidates, Play Video Smooth, Provide Continuity of Video Redistribution and Manage Peer Churn. For instance, in order to achieve “Play Video Smooth” goal, the Peer agent uses PS Managing and Download Managing capabilities. The Peer agent updates its PS List (beliefset in other words) periodically via BM messages and controls its parents every 10 seconds. The agent updates parents’ information by

Parent Scoring behavior. If a problem occurs with a parent (e.g. parent goes offline or video packets are missing due to the low upload capacity), Peer agent will select a new parent using “Buffermap”, “ParentScore” and “SameISP” information in its PS List.

3.3.2. Video Server Agent

Taking into consideration the Video Server agent type, four capabilities are defined as follows: (see Figure 7).

Child Managing: Video Server agent constructs its Child List in this capability. It chooses its children via location information and upload bandwidth capacity while executing its Child Managing behavior. In Child Managing behavior, Video Server agent manages its children according to its available ISP slots. For every ISP, the agent has limited slots. Video Server agent also controls its children’s upload capacity in order to guarantee the video redistribution. If a peer meets these requirements, Video Server agent selects this peer as a child and starts video streaming.

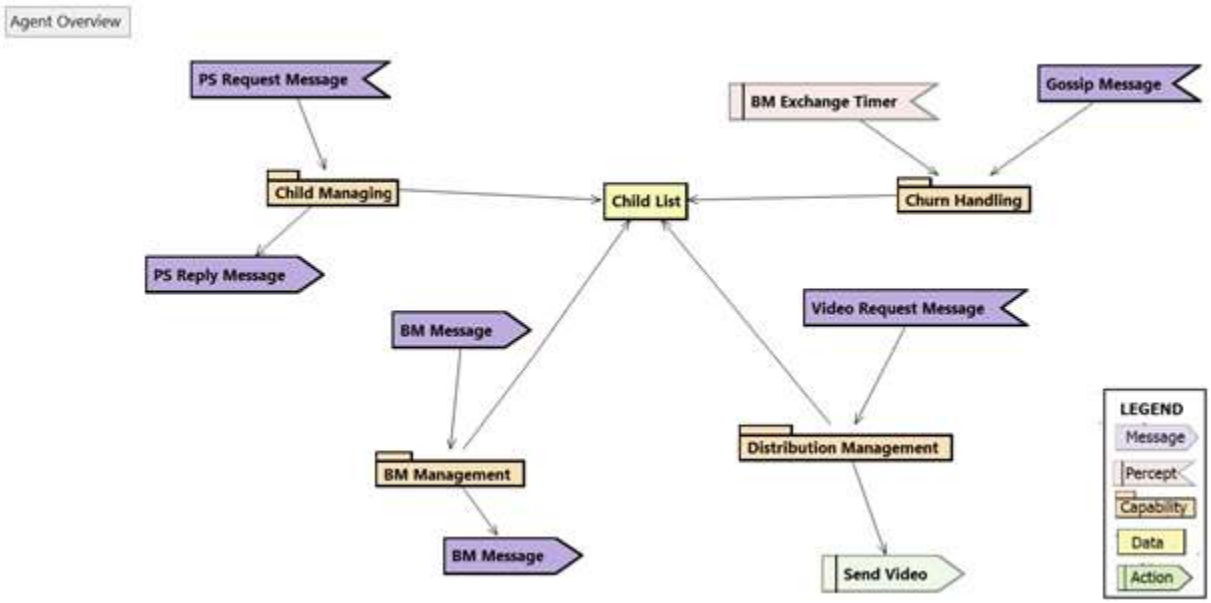

Figure 7. Overview diagram of Video Server Agent

Churn Handling: In Churn Handling capability, Video Server agent controls other peers' online status and updates its Child List.

BM Managing: This capability is responsible from Buffermap Exchange Messages.

Distribution Managing: In this capability, Video Server agent manages video distribution and scheduling. For this purpose, the agent uses the "Child List" which is the list of children connected to the agent. Video Server agent communicates with its children (Peer agents) periodically to inform about the video stream. The structure of the Child List is composed of PeerID, IP Addr, UploadCapacity, ChildInfo and ChildBuffermap. PeerID is a unique ID that is given by Tracker Server and IPAddr is the IP address of peer as usual. UploadCapacity keeps the upload capacity of the peer. ChildInfo is an array of four booleans and keeps child information of substreams. If it is "true", peer sends the substream to the partner. ChildBuffermap is an array of four integers and keeps the information of packets to be sent.

Main goal of Video Server agent is to "Distribute Video in the System". In order to achieve this goal, the agent shows Child Managing and Distribution Managing capabilities. It chooses its children via Child Managing behavior and distributes video packets.

3.3.3. Tracker Server Agent

Third type of the system agents is Tracker Server agent (Tracker agent shortly)As can be seen from the agent overview diagram given in Figure 8, Tracker agent has 3 capabilities:

Peer Welcoming: In this capability, Tracker agent manages newly joined peers and sends MS List to newly joined peers according to its location information and other peers' upload capacities. Tracker agent also sends unique ID to each newly joined peer. All of these actions take place while the agent executes "Peer Welcoming" behavior. Inside this behavior, Tracker agent groups the online nodes and controls whether the ISP of newly joined node and ISP of the selected node are the same. If the nodes are in the same ISP and the selected node's upload capacity is high, these nodes get the high

priority. If the nodes are in the same ISP and the selected node's upload capacity is low, these nodes get medium priority. If they are not in the same ISP, these nodes get the low priority as expected. More information about how this traffic localization is performed can be found in [17].

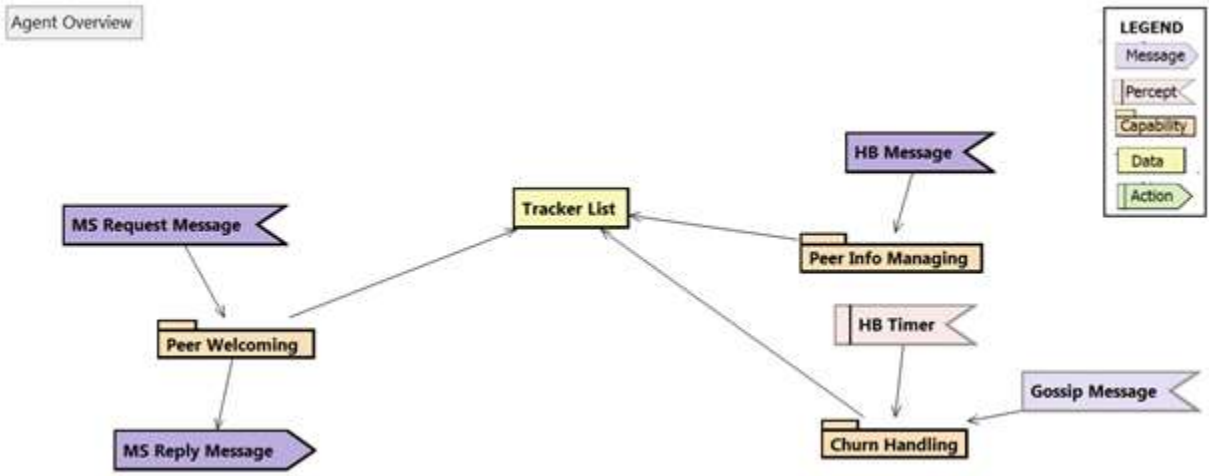

Figure 8. Overview diagram of Tracker Server Agent

Peer Info Managing: In Peer Info Managing Capability, Tracker agent evaluates peers' location information and upload capacity. Peers send this required information via HB Messages.

Churn Handling: In this capability, Tracker agent periodically controls other peers' online status and updates its list when it receives a "Gossip Message" or "Heartbeat Timer" expires. "Tracker List" is the list in question which includes online peers. Peer agents communicate with Tracker agent periodically via HB Messages to inform about the video stream. The structure of the Tracker List is composed of PeerID, IP Addr, UploadCapacity and ISPNo. PeerID is a unique ID that is given by Tracker Server and IPAddr is the IP address of the peer as expected. UploadCapacity keeps the upload capacity of the peer and ISPNo stores the ISP information of the peer.

Tracker agent is responsible from achieving 2 separate goals: "Provide Entrance Service" and "Keep System's Information Up-To-Date". In order to achieve "Keep System's Information Up-To-Date" goal, the Tracker agent uses its Peer Info Managing, Churn Handling and Peer Welcoming capabilities.

Peer agents, which are online in the system, must send HB messages for every 5 seconds. If the Tracker agent does not receive heartbeat message from a Peer agent (i.e. Peer 67) for 5 seconds, heartbeat timer triggers the execution of the tasks defined in Churn Handling capability and the Tracker agent erases Peer 67 from the tracker list. That means Peer 67 is not online anymore.

4. EVALUATION

In this section, we discuss the performance evaluation of our MAS by taking into account its comparison with a CoolStreaming-like system. For the comparison with our agent-based video streaming system, we implemented a CoolStreaming-like system which represents a regular video streaming environment that is popular and currently used in the related research field. It does not involve any agentified components or agent capabilities. Furthermore, traffic localization and upload bandwidth of the nodes in the system are not implemented in agentified parent selection process. It is also worth noting that the performance of the proposed MAS has been first reported in our previous work [18]. However that work only covers the sole performance of the system in various network sizes.

Both CoolStreaming-like system and the proposed MAS are tested in ns-3 environment. ns-3 [14] is a very famous discrete-event network simulator, implemented in C++ and especially used for research and educational purposes.

Simulations are run for a MAS with 300 agents (nodes). The network topology is generated by BRITE [19]. The simulation duration is 900 seconds. There is a video server which distributes the video in the system. There is also a tracker server which owns all network topology information. There are 4 different regions (ISP) in the system. In each region there are 75 nodes. Tracker server groups these nodes according to their ISP information and upload capacity. More information about the topology can be found in [17]. Number of substreams is 4 and each substream bitrate is equal to 75 kbps (total video bitrate is equal to 300 kbps). All the packets are sent over TCP. The bandwidth distribution of all agents in the system is given in Table I.

Table I. Bandwidth distribution of the agents in the simulations

Percentage	**Upload Bandwidth**
10%	*<100 Kbps*
50%	*< 300 Kbps*
90%	*< 1000 Kbps*
100%	*< 5000 Kbps*

The simulation results are given in the following. Since continuity index and startup duration are important for quality of experience (QoE) and P2P systems cause extra load on network backbones, we give the results in terms of continuity index, startup delay and network traffic amount.

Average continuity index values of our MAS and CoolStreaming-like system are given in Table II. Above discussed parent selection, Peer Welcoming and Child Managing behaviors performed by the agents make the performance of the proposed MAS higher than the CoolStreaming-like system.

Table II. Average continuity index values

	Continuity Index
CoolStreaming-like	87.7%
MAS	92.2%

In Table III, average startup delay of the MAS and CoolStreaming-like system are given. Startup delay is the duration which a Peer agent waits until the video starts. Since the CoolStreaming-like system does not consider the network locality, the average startup delay is about 21 seconds. On the other hand, in our MAS, a network locality capability is implemented on the Tracker agent and hence startup delay decreases about 5 seconds.

Table III. Average startup delay values

	Startup Delay
CoolStreaming-like	21.2
MAS	16.1

Finally, the percentage of traffic over the network is given in Table IV. The CoolStreaming-like system does not have any information about the network topology. So peers select their partners randomly and this increases the traffic on the backbone. On the other hand, the Tracker agent of the proposed MAS possesses the network information and performs the Peer Welcoming behavior. Thereby, Peer agent selects its partners from the same ISP and hence Inter-ISP traffic is decreased. While the Inter-ISP traffic of the CoolStreaming-like system constitutes 54% of the overall traffic, the Inter-ISP traffic of the proposed MAS constitutes only 9%.

Table IV. Percentage of traffic over the network

	Intra-ISP Video	Intra-ISP Control	Inter-ISP Video	Inter-ISP Control
CoolStreaming-like	34%	12%	36%	18%
MAS	64%	27%	5%	4%

5. RELATED WORK

Taking into account the use of software agents in P2P video streaming systems, we encounter a few studies in this young research field. Pournaras et al. [7] propose a model to build robust tree topologies. In tree-based topologies, node departure or failure can cause serious problems in the system. Agents are used to fix these problems in their study. But the proposed method cannot be directly implemented to mesh-based systems since tree-based overlay architecture is different from mesh-based overlay architecture.

Carrera and Iglesias [8] present an agent-based design for the diagnosis of multimedia streaming faults. The agent detecting an unusual symptom such as quality degradation chooses to perform the best action. However P2P streaming-specific actions such as parent selection are not addressed in the work. On the other hand, Molina et al. [9] propose a P2P video file sharing architecture for mobile networks. In P2P file sharing approaches, the time of video packets delivery does not directly affect the system performance while the timing constraint in P2P live video streaming systems is crucial.

Orynczak and Kotulski [10] propose an agent-based infrastructure for real-time applications. In this infrastructure, agents analyze parameters like link bandwidth usage and number of lost packets and they establish routing table dynamically. Chen et al. [11] use evolutionary games for cooperative P2P video streaming. To reduce traverse links and increase streaming performance, they implement software agents for real-time streaming systems. Menkovski and Liotta [20] introduce an adaptive video streaming system in which agents examine the traffic and make decisions based on the reinforcement learning.

Our work contributes to above studies by discussing the design and implementation of a MAS that supports agent-based parent selection, traffic localization and overlay construction for P2P live streaming. In this study, we also present the internals of the agents which we believe indispensable for the development of such autonomous software. Considering the complementary studies, the inference mechanisms of the software agents described in this paper are given in previous works [16], [17]. Finally, fundamentals of the engineering of such a MAS can also be found in our prior work [18]. However, that work does not cover the behavior mechanism of Peer agents applied for peer selection as discussed in this paper. Moreover, comparative evaluation of the agent-based video streaming approach with non-agentified solutions is not considered.

6. CONCLUSION

In this study, a multi-agent software design for P2P video streaming system is discussed. Both the internals of each software agent in the system (including their behaviors executed for the effective streaming) and the exact MAS implementation inside ns-3 environment are given. Finally, the performance of the proposed MAS is compared to the popular CoolStreaming-like system.

We have investigated that a well-defined software for parent selection behavior can improve the continuity index in P2P video streaming systems. This means the users in the system watch video in a better quality and lower end-to-end delay.

We have also examined that Peer Welcoming (traffic localization) behavior decreases the Inter-ISP traffic significantly. Since the network-awareness level of P2P video streaming applications is not sufficient, these applications cause extra load on the backbone. Additionally, agents having traffic localization behavior provide decrease in the end-to-end delay and increase in the quality of experience. Finally, the software required for the child managing behavior in Video Server agent has also been implemented in this study and we have observed that the proposed system is more robust and more resilient to peer churn comparing with CoolStreaming-like system.

For future work, we plan to implement negotiation mechanisms as first. These mechanisms can be used for bandwidth allocation and video quality adaptation. Additionally, retransmission and path selection alternatives can be implemented to improve performance of video streaming. On the other hand, current system's single point of failure, that can be encountered on the peer maintenance (when Tracker agent goes down) or video distribution (when Video Server agent is unavailable), also needs to be resolved in the future work. Our aim is to investigate the alternative solutions which are feasible to be applied for the elimination of such system failures. In order to keep robustness of the MAS within this perspective, cloning the Tracker and/or Video Server agents at runtime can be an option. Moreover, a solution, in which Peer agents may immediately undertake the duty of tracker or video server in case of these agents' failures, can also be applied. Finally, load balancing for the Peer

agents and system security against the harmful peers are the remaining open issues which also need to be considered in the future work.

7. ACKNOWLEDGEMENT

This work is funded by the Scientific and Technological Research Council of Turkey (TUBITAK) Electric, Electronic and Informatics Research Group (EEEAG) under grant 111E022.

8. REFERENCES

[1] Xie, S., Qu, Y., Keung, G.Y., Lin, C., Liu, J., Zhang, X.Y.: “Inside the New Coolstreaming: Principles, Measurements and Performance Implications”. Proc. IEEE Int. Conf. Computer Communications, Arizona, USA, 2008, pp. 1031-1039

[2] “PPLive”, http://www.pplive.com, accessed December 2013

[3] “PPStream”, http://www.ppstream.com, accessed December 2013

[4] Liu, Z., Wu, C., Li, B., Zhao, S.: “UUSee: Large-Scale Operational On-Demand Streaming with Random Network Coding”. Proc. IEEE Int. Conf. Information Communications, New Jersey, USA, 2010, pp. 2070-2078

[5] “SopCast”, http://www.sopcast.com, accessed December 2013

[6] Wooldridge, M.: "An Introduction to Multi-agent Systems" (John Wiley & Sons, 2nd edn. 2010)

[7] Pournaras, E., Warnier, M., Brazier, F.: "Adaptive Agent-Based Self-Organization for Robust Hierarchical Topologies". Proc. Int. Conf. Adaptive and Intelligent Systems, Washington, USA, 2009, pp. 69-76

[8]Carrera, A., Iglesias, C.A.: "Multi-agent Architecture for Heterogeneous Reasoning under Uncertainty Combining MSBN and Ontologies in Distributed Network Diagnosis". Proc.

IEEE/WIC/ACM Int. Conf. Web Intelligence and Intelligent Agent Technology, Lyon, France, 2011, pp.159-162

[9] Molina, B., Pileggi, S.F., Esteve, M., Palau, C.E.: "A negotiation framework for content distribution in mobile transient networks", Journal of Network and Computer Applications, 2009, 32, (5), pp. 1000-1011

[10] Orynczak, G., Kotulski, Z.: "Agent based infrastructure for real-time applications", Annales UMCS, Informatica, 2011, 11, (4), pp. 33-47

[11] Chen, Y., Wang, B., Lin, W.S., Wu, Y., Liu, K.J.R.: "Evolutionary games for cooperative P2P video streaming". Proc. IEEE Int. Conf. Image Processing, Hong Kong, 2010, pp. 4453-4456

[12] Padgham, L., Winikoff, M.: “Prometheus: A methodology for developing intelligent agents”, Lecture Notes in Computer Science, 2003, 2585, pp. 174-185

[13] Padgham, L., Thangarajah, J., Winikoff, M.: “Prometheus Design Tool”. Proc. AAAI Conf. Artificial Intelligence, Chicago, USA, 2008, pp. 1882-1884

[14] “Network Simulator 3”, http://www.nsnam.org/, accessed December 2013

[15] Rao, A., Georgeff, M.: "BDI Agents: From Theory to Practice". Proc. Int. Conf. Multi-Agent Systems, San Francisco, USA, 1995, pp. 312-319

[16] Sayit, M., Kaymak, Y., Teket, K.D., Cetinkaya, C., Demirci, S., Kardas, G.: “Parent selection via reinforcement learning in mesh-based P2P video streaming”. Proc. Int. Conf. Information Technology: New Generations, Las Vegas, USA, 2013, pp. 546-551

[17] Teket K.D., Sayit, M.: “P2P Video Streaming with ALTO Protocol: A Simulation Study”. Proc. IEEE Int. Symp. Broadband Multimedia Systems and Broadcasting, London, UK, 2013, pp. 1-6

[18]Teket, K.D., Kaymak, Y., Sayit, M., Kardas, G.: “Engineering a Multi-agent System for Peer-to-Peer Video Streaming”. Proc. IEEE Int. Symp. Innovations in Intelligent Systems and Applications, Albena, Bulgaria, 2013, pp. 1-7

[19] Medina, A., Lakhina, A., Matta, I., Byers J.: “BRITE: An Approach to Universal Topology Generation”. Proc. IEEE Modeling, Analysis and Simulation of Computer and Telecommunication Systems, Ohio, USA, 2001, pp. 346-353

[20] Menkovski, V., Liotta, A.: “Intelligent control for adaptive video streaming”, Proc. IEEE Int. Conf. Consumer Electronics, Las Vegas, Nevada, USA, pp. 127-128