Spring 2012

Ad-hoc Stream Adaptive Protocol

Ray Xie
San Jose State University

Ad-hoc Stream Adaptive Protocol

A P2P approach for VANET

Ray Xie

05/01/2012

Abstract

With the growing market of smart-phones, sophisticated applications that do extensive computation are common on mobile platform; and with consumers' high expectation of technologies to stay connected on the go, academic researchers and industries have been making efforts to find ways to stream multimedia contents to mobile devices. However, the restricted wireless channel bandwidth, unstable nature of wireless channels, and unpredictable nature of mobility, has been the major road block for wireless streaming advance forward. In this paper, various recent studies on mobility and P2P system proposal are explained and analyzed, and propose a new design based on existing P2P systems, aimed to solve the wireless and mobility issues.

Keywords: Ad-hoc network, P2P, Streaming,

Table of Contents

Acronyms

4G	Fourth generation of cellular wireless standards
AMF	Availability Management Framework
AODV	Ad hoc On-Demand Distance Vector
AP	Access Point
ASAP	Ad-hoc Streaming with Adaptive Push
CKPT	Checkpoint Service
CLM	Cluster Membership Service
ETT	Expected Transmission Time
DHT	Distributed Hash Table
OSI	Open Systems Interconnection
LAN	Local Area Network
MAC	Media Access Control
MANET	Mobile Ad-hoc Network
P2P	Peer-to-Peer
RTT	Round Trip Time
TCP	Transmission Control Protocol
VANET	Vehicular Ad-hoc Network
VoIP	Voice over Internet Protocol
UDP	User Datagram Protocol

Chapter 1. Introduction

1.1 VANET

VANET (Vehicular Ad-hoc Network) is a kind of MANET(Mobile Ad-hoc Network), which is a decentralized mobile wireless mesh network. Without any central base station, packets must traverse through each mobile station the same way that packets are routed through routers over the internet backbone. Thereby, each station is both router and transceiver at the same time. Using ad-hoc strategy gives many advantages such as giving higher data rate by using short range technology such as IEEE 802.11 or reducing network infrastructure cost because the base station is no longer necessary. However, MANET also suffers heavily on its flexibility. One of the special features in MANET is this: these mobile stations can move, leads to significant problems in term of routing protocol because the path can be swiftly lost without any predictable reason. Nevertheless, MANET is still a popular topic in term of academic research and real world implementations such as sensor networks, 4G cellular systems, military uses. VANET is MANET with vehicles being the mobile device, therefore it is the extreme kind of MANET with fast mobility.

1.2 P2P

P2P (peer-to-peer) is a content distribution mechanism that has grown very popular in the recent years; it relaxes hosts that want to distribute contents by making the downloaders to be part of the supply, this main strength of P2P system is welcomed in almost every kind of everyday file sharing since companies no longer have to worry about upgrading their servers to meet the increasing demand every once a while. A node in a P2P system plays a mixture of roles; it could be a server, a client, and also an intermediate node that relays traffic at the same time. P2P system was mainly used for cheaper and scalable internet content distribution in its early days, and now it has

extended its footprints in the domain of multimedia streaming; Skype is one example that makes use of P2P technologies so for users to share video and audio with each other. In many ways, P2P and MANET share common characteristics and they mainly differs by their position in the OSI (Open Systems Interconnection) seven layers, where P2P resides in the higher layers (session, presentation, and application), and MANET is a concept that focus on lower layers (mainly the network layer). As a result, P2P systems deployment face the same difficulties under the highly dynamic wireless environments [1, 12].

1.3 Streaming

Ever since the introduction of world-wide-web, the internet has been consistently evolving at a rapid pace during the last couple decades; from the classic days of web 1.0, where the web was mostly a "read-only" database that users mainly only retrieves information from, then to the boom of web 2.0 (also referred as "read-write web"), where the breakout of blogs, social networking sites, and interactive sites open ups the doors for user to collaborate with each other, and make contributions by publishing contents or leaving feedbacks; then to the dawn of web 3.0 (semantic web, also referred as "read-write-execute") that emphasis on a web that "understand" the user and contents, which knows how to process the web data and related them to the user tasks. Multimedia content might be rare in the days of web 1.0, but not anymore after entering the age of web 2.0 with more and more TV programs being broadcasted (live or on-demand streaming) over the internet; and growing number of people replace their traditional landlines with VoIP services such as Skype and Vonage. Although streaming technologies exists for quite a while, but new challenges surface as user making higher and higher expectations out of it. Being able to delivery multimedia content in a timely but also secured manner is never an easy task, and academic researches and the industries has spent a lot of efforts in the past decade to achieve a public

entrustment in VoIP (Voice over Internet Protocol) and video chat services, but now it face the same challenge again with the emerging 4G cellular and wireless mesh networks and the powerful personal mobile device such as smart-phones [13, 19], user's demand for on-the-go streaming is growing greater each day.

1.4 Ad-hoc wireless P2P streaming

When we put VANET, P2P, and streaming together, we might able to find a solution to the difficulty of handling video streaming that some cellular service providers are having with it's already overloaded data network. With the recent powerful smart-phones, which are capable of making video calls with high clarity, some subscribers still cannot take advantages of those cutting edge features the hardware provides mainly because of the bandwidth problems the cellular data network is having. VANET and P2P together could be a perfect match to solve the limited bandwidth problem as VANET provides a way for these mobile devices to dynamically interconnect to each other with their Wi-Fi capability, and P2P allows them to distribute and share contents using those connections. In an urban neighborhood with high population densities, and given the common ownership of a smart-phone or a Wi-Fi hotspot in each household, it's possible to create a VANET that covers an entire urban city; with the advantages of being self-maintained and self-scale using VANET and P2P, bandwidth would no longer be a road block of making video calls.

Although putting VANET and P2P together can solve the bandwidth limitation problem, it also creates one problem to be solved in order to make it practical. VANET and P2P suffers from mobility issues [12, 13, 15], mainly link failures that involves route replacement lookup and connection reestablishment, and that ultimately lead to overhead in transmission time, which is a big impact on the user perceived quality since video and audio streaming are highly time sensitive. The current P2P pull-based systems that rely heavily on steady connections will not work, and

therefore a proposed new P2P system, namely ASAP (Ad-hoc Streaming with Adaptive Push), will be covered in the paper, with five objectives: 1) Limit the bandwidth consumption on P2P Streaming sessions to a satisfactory state as much as possible, 2) Aggressively fill-up the initial buffer to start playback for increased user-experience, 3) Conserving the bandwidth by avoid any unnecessary "future" video segment downloads that could render useless, as smooth stream require consecutive segments, 4) Able to react quickly to topology changes such as route changes, 5) Limited to application layer implementation such that deployment is at ease.

Chapter 2. Related studies

2.1 P2P systems in wireless environments

2.1.1 Impact of mobility

Moraes et al [12] conducted an experiment in their study to examine the performance of various popular P2P streaming systems under wireless environments. Some of the most popular P2P streaming protocol selected for experiments includes: SopCast, PPLive, TVAnts, PPStream, and ESM; ESM is the only protocol that's tree-based, the rest being mesh-based. For transport protocol, each of the application has different preference: SopCast uses mainly UDP (Transmission Control Protocol), and with a few TCP (User Datagram Protocol) for control messages; PPLive uses TCP via few UDP; TVAnts use both TCP and UDP equally; PPStream uses only TCP; and ESM uses TRFC.

Tree-based P2P overlay is one type of structured P2P that enforces a certain type of structure of the logical network topology formed by peers. In ESM, the root is the video source, and nodes can receive from the upper tree (which includes root, parent and sibling nodes). The advantage of using tree-based overlay is that it has low latency and low control overhead when topology changes are rare, but the flip side is that the tree reconstruction is very expansive when peer connections change frequently.

Mesh-based P2P overlays offers freedom to the topology structure since there is no structure enforcement; peers can connection to any peer in the overlay. In meshed-based P2P, data is divided into small chunks, and peers will exchange chunk availability information with a selected subset of peers, so peers can download chunks from multiple peers concurrently. The good side of mesh-based P2P is being less sensitive to node departure and link failures, but the disadvantage is it requires additional overhead for exchanging chunk maps.

The study shows that TCP-based application (PPStream) is most sensitive to gateway changes as peers roam between different coverage areas and the data delivery will suffer from gateway changes. Video playback may just freeze as data transmission sometimes drops to zero due to congestion control, and the bidirectional nature of TCP cause performance degradation occurs on both downstream and upstream. On the other hand, UDP-based application (SopCast) is less sensitive to gate way changes even though data delivery still suffers from gateway changes, video playback does not freeze as data transmission does no drop to zero, and the upstream traffic is only minimally impacted by gateway changes as there is no feedback as in TCP.

The authors found that both TCP-based and UDP-based P2P streaming applications benefit from short-lived connections. UDP enable a smoother adaption to Wireless Mesh Networks, and TCP-based application is also feasible if numerous short live connections are used. For this reason, our proposed solution chose to use only UDP for its transport layer.

2.1.2 P2P streaming evaluations

In another research conducted by Chan et al[1], various P2P lookup cores are compared under wireless environments, the study included the three common DHT (Distributed Hash Table) lookup cores: Chord, Kelips, and Tapestry. In the nutshell, DHT is a mechanism that allow nodes to efficiently route a search query by mapping data blocks to keys using the data block's unique attributes; DHT assigns each peer with a set of IDs of the data blocks, and each peer also gets a unique ID, and the look up will return the ID(s) of the node(s) that has the desired data block.

Chord uses consistent hashing to map peer and data into m-bit identifiers, where the peer ID is a hashed value of the peer IP, and the data block ID is a hashed value of the data attributes. The logical network topology in Chord forms a ring by connecting peers clockwise from smallest to

largest, also known as Chord Ring; a peer in the ring is responsible for storing data IDs' ranging from the first data ID that's greater than its counterclockwise neighbor's ID, to the last data ID that is less than its own ID. Each peer builds an internal finger table that points to peers in the clockwise direction with the exponential increase for the ID of the peer the entry points to; this finger table is mainly built for more efficient lookup that can jump through intermediate peers more rapidly. Each node also contains a successor list which is use for periodical confirmation of the successor-predecessor relationship with its successor. These two local structures are periodically updated using the stabilization process. With the formed ring and the local knowledge, Chord is able to perform the lookup by forwarding the query to the peer with the biggest possible ID that's still smaller the desired data ID, until an owner of the file is reached.

Kelips uses a very similar concept as Chord, but the mapping is quite different than Chord. Kelips divides identifier space into k groups, which k is approximately the square root of the number of nodes; then each node is assigned to a group whose group ID is the peer's ID mod k. Data blocks are also assigned to a group by hashing the meta data. Each node contains a group view, which includes all other nodes that belongs in its own group, plus a few external contact nodes for rest of the groups. On top of that, each node also builds a table of mapping of the data block to the owner's IP, but the table limits to the data blocks that belong to the same group as the peer's group. With all these information, Kelips forwards queries based on the group ID of the desired data block, if the data block belongs to a different group than the peer, the peer will just forward via the information about the contact nodes in that group, if the data block is in the same group, then the peer will just forward the query to the owner.

Tapesty uses a very different mapping mechanism other than Chord and Kelips but very similar to CIDR since it's prefix based. Tapestry structure its identifier space as tree, the route table

at each node is split into k levels, and the nodes in the kth level share a same prefix of k – 1 length, and each level contains a primary contact and a backup contact, sorted by latency. Then lookup queries are forwarded to the longest match on the prefix, where closest match is used when no exact match is found.

The study found that under high churn rate, which means the high rate of peer join and departure, Chord performs the worst, while Kelips being medium and Tapestry the best; for medium and low churn rate, Chord actually perform the best, and Kelips being medium and Tapestry being the worst.

In addition, authors in [13] and [18] conducted studies to find out the most influential factor in maintaining an efficient P2P streaming system with good level of supply, on both live and on-demand systems. It turns out that the average peer contribution is linearly dependent on the upload bandwidth of each peer, and is independent of the arrival rate, file duration, and neighborhood size. In ad-hoc wireless networks, with high peer churn rate, in order to maximize the utilization of upload bandwidth from peer, a data delivery mechanism that can adopt quickly to route changes is critical.

2.1.3 Duplicated content delivery

Rodrigo et al [14] in their research studied and stimulated mesh-pull based P2P streaming traffic, and conclude that these P2P systems are wasting bandwidth on delivering the same content for different end receivers, and this found that the wasted bandwidth is close to 30% of the total traffic. Simulations also found that the problem could be remedy by localizing peers and providing cache entities into the system. These findings again proves the major waste in P2P systems are duplication, thus reducing duplicate sends is another key in boosting performance.

2.2 P2P VoD streaming enhancements

2.2.1 Priority based scheduling

Chen et al [2] in their paper proposed a priority based scheduling for P2P VoD content retrieval that is a combination of three priority scheduling: FPO (Fluent Playback Oriented), that strike to retrieve continuous block of segments followed the playback offset; SAO (Service Availability Optimization), which try to increase content availability by pushing off content to selected peers with enough capability; TMO (Throughput Maximization Oriented), which try to maximize the throughput by pre-fetching segments ahead of playback buffer until it reaches its download capability, which is not a suitable feature for VANET environments since it is desirable not to pre-fetch segments to be conservative on bandwidth consumption.

FPO uses an additional buffer that marks a certain number of consecutive video segments that following the playback, then the video segments inside this buffer will be retrieved with higher priority other than the rest of the stream. This prioritized retrieval has a simple goal to make sure the playback is smooth by downloading those segments have a sooner playback deadline.

SAO is a simple idea that keeping track of the availability of a video segment, when the availability drop between a certain threshold, then the pusher (the peer that contains the segment) will determine an optimal candidate to receive the segment; the determination criteria includes: the upload bandwidth, the larger the better, as the candidate will be a stronger server; the current playback position, smaller than the segment being pushed but the closer the better, as the candidate will more likely retrieve the same segment for its playback anyway; the residual lifetime, the longer the better, as we want to keep the increased availability as long as possible. In order for SAO to work, peers need to sent periodical information such as demand/supply ratio of segments

back to the information layer. This periodical information exchange is not desirable in VANET as it would create too many chatty clients, and peers detach and reattach to the network very frequently, a peer departing the network might only have temporary effect on the availability.

2.2.2 Caching

Dai et al [3] proposed a simple caching algorithm for storing video segment blocks at peers. The proposed solution simply dived the peers into group clusters, and each cluster maintain a mapping that maps a segment cache to its owner; and it uses a simple popularity based replacement policy such that the least used cache segment will be replaced first. This inspired the need for caching in our design, although we will be simply using a sliding window for replacing policy.

2.2.3 Mathematical approach

He et al [5] used mathematical approach to solve capacity problems, by first putting bandwidth capacity into equations, solving the equations to find the parameters that will yield optimal bandwidth; the paper proposed increasing streaming capacity, in a sense of server bandwidth, with introducing helpers into the equations (which really is just asking more peers to do retrieval to increase content availability). The paper is strong in terms of its mathematical theories and equations, but it does not apply to our problem that trying to limit our bandwidth usage on the network, since adding helpers means adding bandwidth usage to the physical layer, it only helps to increase capacity on the application layer since there is higher availability of the content. Things that learned from this paper are the important parameters in a P2P VoD systems, which are our main measurements for our solution's efficiency.

2.2.4 Push-pull based P2P multi-streaming with multicast

HyStream is the proposed design by Liu et al [9] in their paper to enhance the performance of live streaming to multiple receivers; it uses a simple idea of combining the existing push model being used in tree-based P2P streaming system, and pull model used in mesh-based P2P streaming system. The push model is based on the DHT (Distributed Hash Table) overly used in most centralized P2P system, then nodes join multicast trees that the root node will push the content down to the child nodes, which is also known as push-tree. The pull mechanism is only used to remedy the weakness of slow tree recovery from node failures. The study results shown improvements over the push only models.

The push model used in HyStream inspired ASAP initially designed to use a push model for its data delivery; whoever the performance turns out not meeting our expectation due to the same reason of the expensive overhead in keep the serving node synchronized with its receiving peers, especially when ASAP is targeted at VoD instead of live streaming the Push model used here, the cost of synchronization between is even more.

Nonetheless, the demulitplexing feature of multicasting is used in ASAP, through a different mechanism, to reduce the chance of sending duplicated data downstream to the receiving nodes from the serving node.

2.3 Wireless P2P streaming for cellular networks

2.3.1 Utilizing access point P2P agent

Lai et al[7] in their research proposed an improvement for wireless P2P streaming in cellular network. The proposed solution make use of the not only the peers but also access points (base stations) and dedicated P2P source servers, to help build the P2P overlay and forms a wireless

community. In order for the solution to work, specialized P2P software is required to be installed on the wireless terminals and the access points, and the solution uses a multicast delivery paradigm. The P2P source server will host the original source of the data, and P2P access points can receive and relay data, while the P2P wireless terminals remains the same as a regular P2P node.

The solution adopt SCAMP (Scalable Membership Protocol) using gossip-based mechanism; each receiving peer owns a partner list that includes the identifiers of the partners that server to receiver, known as PartialView, and each sending peer keeps a consumer list, tracking the nodes that are receiving segments from it, known as InView. A wireless interested in joining a multicast group will send a join message to the P2P capable AP, the AP then will sends a subscription request to the contact node which is an arbitrary member, and adds the contact node to the PartialView, then the contact node will forward the joining node ID to its entire PartialView, and a number of additional randomly chosen nodes, then a node receives the subscription request will decides whether or not to add the node to its PartialView with a probability P = 1 / (1 + current size of the PartialView), if the subscriber is not added, then the subscription will be forward to a random node chosen from the PartialView. The PartialView entries will expire if a corresponding peer does not re-advertise within the timer expiration interval.

The study claimed that the solution expands the amount, speed and reliability of the data transmission compared to existing P2P overlay that limited sending/receiving function to peers only, therefore it introduce smoother video streams with less jitter. The problem with the study is that the experiment was setup with a dedicated serving node on the “faster” LAN (local area network) side of the setup, which violates the “all nodes are equal” P2P semantic, and the solution might not be beneficial for scenarios that the serving node exists in the wireless part of the network, which is very likely the common scenario in a large scale decentralized P2P system. The

solution might work given that there is a P2P capable agent, a dedicated sever in the faster backbone internet, and limited to live streaming.

2.3.2 Energy-efficient P2P live streaming

Li et al[8] in their research proposed a solution that balances the three goals of: 1. minimizing playback buffer underflow to improve user experience; 2. limiting bandwidth usage; 3. achieve energy conservation. The solution made quite a few assumptions and requirements; a somewhat more powerful peer would be randomly selected as a coordinator for a group of peers, and this coordinator will perform some critical assessments on individual peers to make optimal delivery decisions for every scheduling interval of T; these assessments consider the availability of playback buffer, channel gain between any particular pair of peers, and willingness to consume battery.

On the receiving side, in order to improve the playback smoothness, the proposed solution maximize the available continuous playback buffer by always making the coordinator to choose the peer with the least amount of playback buffer to be the node immediately served.

On the sending side, there are a few algorithms proposed with small variation in each of them. The first step in all the algorithm is of course to locate the node that can serve the requester, once the servers are found, the coordinator has pick a sever among multiple servers; algorithm 0 selects the peer with least content availability to serve content first, the idea is to save energy of the other peers with higher content availability to serve rarer content later; algorithm 1 on the other hand selects the peer with the best effective transmitting rate to be the server, the measure used could be channel gain. There are two extensions to algorithm 1, one is consideration of the willingness to use battery, that weights the transmission effectiveness by the willingness to use

battery, which is proportion to the remaining battery life; another one is allowing broadcasting, which requires the coordinator counts the number of repeated request of the same data segment, then prioritize to send the segment with most repeated request, then chooses the peer with the highest effective transmitting rate to receive the segment and broadcast it.

The authors claimed that the solution achieved up to 190% energy efficiency, however, the results does not mean the solution is workable since the assumptions made up front was too ideal to start with, and in order for the coordinator to make decisions using the measures proposed, there will be large amount of data collected and processing of these data take power too, these overhead is not considered in the study.

2.3.3 Smart caching

A caching solution was proposed in the research conducted by Tan et al[16], the idea is to save bandwidth via caching the stream data at the AP to assist upstream traffic, so to reduce the bandwidth usage when serving nodes supplies data to the receiving nodes. It is common in P2P systems that receiving nodes will be serving the same data block soon after it completed the retrieval, the uploaded data could be a much as ten folds the downloaded data, this nature of P2P node playing the fair game of being both part of the demand and supply creates traffic load on the wireless channel that could lead to congestion; this caching solution also the wireless terminals to just send a compact meta data that represents the cache at the AP (Access Point), then the AP will replace the meta data with the cached content, then send it to the receiver.

In the smart caching scheme, both the AP and the wireless terminal will need to keep a downstream data buffer that stores the recently downloaded data segment, the buffer at the AP will need to be partitioned into multiple buffers that assigned to different wireless subscribers.

When a serving node want to send some data to the receiving nodes, it will first perform duplication detection comparing the data it wants to send to its downstream data buffer, if duplication is found, it will replace the original content with a mapping value prior to sending out the data. The duplication detection could be done via either a hashing function or a more efficient method called Rabin Fingerprinting, the advantage of fingerprinting is it allow distributive addition, meaning each additional segment could be mapped independently and add the values to the previous mappings to get the new mapping that represent all the segments.

In order to achieve high cache hit rate, both the buffer at the AP and the wireless terminal need to be in-sync; whenever the buffer is out-of-sync, the cache lookup will have a chance to fail when the compressed the data packet arrived at the AP. This case is not rare since the time it takes to perform the duplication detection, compressing the altered data packet, and finally delivered to the AP might have been long enough for the AP to receive enough new data segments that just pushed the matching cached content out of the buffer. In this case, the authors used a simple solution similar to duplicate ACK's used in TCP, such that the original packet will be sent when the AP fail to find a matching content and requested the same compressed packet for three times.

The performance evaluation given by the authors was impressive such that the duplication detected and reduced is as much as 71.82% using fingerprinting, and 22.25% using hashing, for PPlive, 86.60% and 34.83% respectively for TVAnts, 85.94% and 89.17% respectively for ESM. However, this solution is high application dependent, since it relies heavily on the fact that the received content is soon to be served, the reason for the gain variation in those three different applications is their different behavior in terms of how soon the received content is being served.

This caching solution could be further enhanced by caching two-way data, meaning both the upstream lookup and the corresponding downstream replied data, and keep them associated at the

AP, the buffer out of sync problem could be completely eliminated by using this scheme since the AP has enough knowledge to generate the response, since lookup queries are relatively small comparing to the responded data, this means small overhead at the AP, but that offers a big benefit of not have the less powerful wireless terminals to have buffers and perform duplication detection on the data since the duplication detection could be done by just examining the request.

2.3.4 Cloud assisted media streaming

Jin et al [6] paper proposed offloading the bandwidth burden to co-located peers with an additional data center in the cloud that tracks peer statistics. The proposal called for data center that tracks peer correlation that keeps a profile for each peer (what other peers it has met and how long they last together in the same zone), using that statistics to predict and select peers to help content retrieval and deliver the content locally. Our proposed has a similar strength such that it make use of locality for providing extra bandwidth, but it has the advantage of not using any prediction for optimal peer selection, it does what it could do the best at the moment.

2.4 Wireless P2P streaming for mesh networks

2.4.1 Unified P2P and cache

Zhu et al[19] in their research also proposed a caching scheme for mesh networks, this solution simply adds assisting entities into the mesh network to help streaming data, and it does not change the original P2P implementations.

Mesh content servers are added throughout the mesh network, the key is to place at least one mesh content server near every access point, so that the wireless terminal can reach at least one mesh content server within a few hops. The content server will cache the streaming data and serve them to requesters. Assuming a main content server contains the original source file, the

main content server will deliver the source file to the cache content server during off-peak hours so the mesh content cache servers are pre-populated with the contents.

When a client wants to retrieve data, it will select two cache content servers, one being the primary and another being the backup. Mesh client will fetch the first N number of clips from the cache server to minimize the startup delay, and then the client will try to fetch the rest of the content from the peers to utilize other network resources and balance the load, but if a clip cannot be downloaded from a peer before its playback deadline, the mesh client will fetch directly from the cache server. The mesh access also need to implement proxy capability that will collect path cost to source peers and inform the requesting peer the costs, then the requesting peer will select peers under the same access point or better path cost.

The proposed solution offers two different ways of discovering mesh content cache server: 1. A centralized index server will perform selection of the primary and backup content server for a requesting peer, such that it will select he servers with least load or least number of peer that the server is serving; with the centralized index server, the requesting peers will not need to have knowledge about load, but the cache servers will need to periodically report to the centralized index server. 2. The requesting peer floods the network with a mesh content server request message for a particular clip; potential servers received the request will send back replies with the hop count or the ETT (expected transmission time) from the routing protocol, the requesting peer then will choose the server with least number of hops or the lowest ETT.

Authors claimed that the solution increased the capacity of content services, meaning the system can serve more clients at the same time, and it improved streaming quality and workload posed on the servers and networks. However, this solution has a serious flaw that it assumes pre-populated cache servers, the delivery of the cache content to these servers still poses bandwidth

consumption, and it is impractical to make to content server to store every possible stream that peers are going to request.

The solution might be practical if the cache content is populated based on the demand, such that the cache is being populated as peer downloads them. Peers can select their primary and backup cache server upon joining the network, and then always sends content request through the cache server; if the cache server has the content, it will act like a serving peer and reply back to the peer and help to forward the request to the rest of the network, on the other hand, if cache is not in the cache, the server can act like a requesting peer on behave of the requesting node, such that the server can download and cache the data.

2.4.2 Redundant tracker

Liu et al[10] in their proposed a solution to improve mobile P2P streaming service with redundant tracker. The idea is as simple as adding a backup tracker to the P2P system so to introduce higher availability of the tracker, which is the critical part of the system that uses a tracker since it is responsible for bootstrap a joining peer to discover the P2P overlay.

The proposed design of a tracker consists of three major modules: AMF (Availability Management Framework), CLM (Cluster Membership Service), and CKPT (Checkpoint Service). The AMF is responsible for selection and designation of Active/Standby role of a tracker, since there are multiple trackers in the system, each of these nodes can be either in Active mode, which will be servicing the bootstrapping process, or Standby mode that it does not involve in bootstrapping process until it is selected; AMF will be monitoring software components and relevant resources, and handles fail-over to provide uninterrupted service. CLM will detect external failures such as member departure, it uses heartbeat mechanism to detect failure and immediately notify all cluster

members; CLM is also detecting new tracker when it joins the cluster. CKPT is the module that will keep trackers in sync, the active trackers need to use the module to create check point data and push to each of the standby trackers.

Authors did not present a valid performance evaluation to support the claimed improvements, the results only show the performance of the proposed tracker, and there were no comparison to performance of existing tracker designs. In addition, using tracker-based P2P system is not suitable for wireless ad-hoc environments as tracker needs to be offer high availability if not 100% up time, which is what peers in an ad-hoc network lacks.

2.4.3 Maximum utility peer selection

Gurses et al[4] in their study proposed a complicated cross-layers design that monitors the underlying wireless channel conditions and select an optimal TCP connection for streaming. Traditional P2P streaming select source peer randomly leading to sub-optimal throughput, and the lack of load balancing could also lead to congestion even when optimal source peer is selected; the slowdown of the network affecting other applications' performance. The solution proposed by authors involves extensive mathematical equations and theories, which will not be explained in this paper. The solution basically monitors each established TCP session and select the optimal TCP session adaptively based on transmission rate, and also make use of MAC (Media Access Control) layer information to calculate collision possibility and channel quality. Whenever a peer wants to send data, it will perform the two evaluation, and select the channel that has the lowest collision chance, the best quality, and the highest observed transmission rate (hence the name "maximum utility").

The performance evaluation shown improved overall network throughput, but with lower P2P through yields higher throughput for other unicast flows. By considering the chance of collision and yielding to other consumers of the wireless channel, the solution is a friendly P2P system that make use of connection that are free first, and that does not guarantee higher throughput for P2P, only a better balanced network that could increase the overall throughput and make P2P to better co-exist with other applications in the network. This solution is unnecessary complicated to achieve such a simple goal; the same goal could be achieved by simply capping P2P stream rate to a satisfactory state and choosing peers that responded with the lowest RTT (Round Trip Time).

2.4.4 Cross-layers optimization

Luo et al[11] proposed another cross-layer optimization for wireless P2P streaming. The cross-layers design is targeted to tackle the weakness of traditional P2P systems that assume perfect conditions in the lower layers. It makes use of the state information flowing through the protocol stack to adapt their behaviors accordingly by using a controller. At each layer, key parameters are monitored and adjusted, including the following: quantization step size and prediction mode at the application layer, which will affect the ability of the receiver to recover from corrupted bits; frame size at MAC layer, which could lead to disassembling if it is too large or assembling if it is too small; and modulation and channel coding at physical layer, which affects transmission delay.

Whenever a requesting node wants to retrieve a segment, it will send out a request to the suppliers, and each supplier will perform an estimation of the video distortion (when the segments arrived at the requester side) with its optimal cross-layer parameters; then the requester will choose the supplier that offers the least distortion, when either all responses from suppliers are

received or a timer expires, the timer is dynamically calculated based on the playback deadline of the requested data.

The study shows that the user perceived video quality is higher due to the higher successful rate of retrieving the data within the timing-requirement (playback deadline). However, this solution is also too complicated due to its cross-layer nature, and it lacks the consideration of mobility issues where receiver or sender can be disconnected from the network at any time, leading to the lack of fallback mechanism when the sender is no longer available. On top of that, the estimations are perform by the suppliers at the time they received the request, and that does not guarantee the same conditions will remain until the completion of the data transfer, which lead to inaccuracy of the estimate; always choosing the best sender would also mean abusing the best sender.

The solution can be improved by having the receiver validate a sender's distortion estimation after the transfer is completed, calculate and compare the actual distortion to the advertised distortion from the sender. As long as the discrepancy between the actual distortion estimation and the actual one falls within a defined threshold, or the data is received within the playback deadline, the requester does not poke for estimation again for the same sender for the next request. With the validation, the receiver can also build credit profile for each of its senders and use the credit profile to weight their future estimations.

2.4.5 Network coding

Wang et al[17] in their research applied network coding to help reducing redundant traffic generated by serving the same video data content over time. The key idea of network coding is to allow intermediate nodes to perform local decoding of incoming packets and encoding of outgoing

packets, thus offering intermediate nodes the ability to repackage packets into a single transmission, then broadcast or multicast the content to intended receivers.

For network coding to work, All outgoing packets are encoded using randomized encoding vector locally, and a global encoding vector is included in every encoded packet so any receiver can decode it given the local encoding vector is random and the field size is sufficiently large. Adding the global vector to every packet introduces overhead but eliminates the need for topology knowledge to compute encoding and decoding functions. Practical implementation of network coding at network layer requires buffering of packets at intermediate nodes as packets will arrive out of order, but encoding requires the data vector to be consecutive.

With network coding, there is less redundant traffic in the network when demand outgrows supply and able to maintain smooth playback under high peer dynamics. But there is one serious weakness of the solution is that the accelerated implementation presented by the authors requires specific set of x86 SEE2 instructions, which is needed to have negligible impact for the encoding and decoding process.

Network coding applied in the proposed solution is really just splitting video segments into smaller chunks, and requesting chunks from multiple seeds, the benefit of sustainable smooth playback under high peer dynamics might just be a side effect of serving smaller chunks of the segment (leading to shorter transmission), and retrieving them from multiple seeds (single source failure tolerant), and the reduction in redundant traffic is caused by smaller size chunks being lost lead to smaller retransmissions.

2.4.6 Caching

Salta et al [18] proposed using a cache scheme to improve P2P streaming performance. In their study, authors simulated a common P2P scenario in wireless mesh network, where contents are being deliver over the same sub-routes to different end points. The paper propose to solve the problem by capturing and cache P2P traffic at each peer, then serve from the cache for multiple requests. Results shown that duplicated traffic dropped to 1.3% from 46.7% of the total traffic. This paper once again prove the benefits of using a caching scheme.

2.5 Observations of proposed solutions

Many of the proposed design to improve wireless P2P streaming lack the consideration of mobility issues, such that the wireless roams between different coverage areas. The experiment results shown in [7] indicate that the streaming session suffers mainly because of gateway / AP changes. The proposed solutions all try to solve the issues introduce by wireless environments by either adding assisting entity into the system, or use some monitoring mechanism to observe the wireless conditions and try to predict and select the best path possible. Adding entity is not an option for ah-hoc wireless network as the network is build with solely the wireless peers, and any monitoring and prediction scheme would bring very little benefits given the high peer dynamics in ad-hoc wireless networks that is just a big cost for monitoring overhead with a small return.

In the next chapter, a push-based solution, ASAP, that try to take a different approach to make P2P Streaming feasible in wireless ad-hoc environment will be described, which is inspired by the simplicity and effectiveness of AODV (Ad hoc On-Demand Distance Vector), which does not monitor, predict and select, it just simply perform the task on the fly, and use a fast recovery mechanism to recover from link failure. ASAP also strike for simplicity and uses a fast feedback

mechanism to adjust the P2P streaming session adaptively, instead of try to pick the most stable wireless channels and rely on the accuracy of the prediction, it just simply embrace the unstable channels with a simplified pull model and uses only UDP.

Chapter 3. ASAP

3.1 P2P characteristics

ASAP is a software solution that sits in the host layers (OSI layer 4-7) just like any other P2P system, as depicted in Figure 1; it could be deployed to any kind of network with different combinations of the lower layers. As for transport layer, ASAP chooses to use only UDP to reduce the side effects of often route changes in VANET.

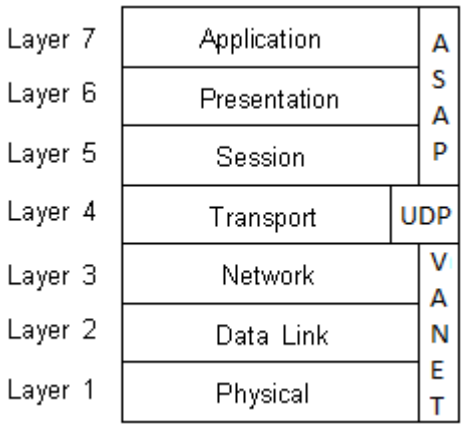

Figure 1. ASAP Architecture

ASAP inherits the fundamental P2P concept of data management, it splits the video stream into equally sized small chunks, which will be referred to as segments for the rest of this paper. Each data transmission will contain at most one segment, and encapsulated in a ASAP message format following the ASAP message header that contains necessary metadata ; each of the segments will be numbered, and these numbers are used as unique identifiers of segments.

A decentralized P2P system offers complete freedom to node insertion and departure to the network, and therefore a freshly joined node has to rely on a flooding scheme to locate its peers and make the peer-to-peer connections. ASAP uses the same concept of broadcasting request for its discovery phase of finding out serving nodes in the network. However, it uses solely UDP that

requester does not initiate TCP connections back to the serving peers like many other traditional P2P systems. Furthermore, ASAP limits requester to retrieve content from one serving peer at a time, therefore there is no peer selection at the requester; the requester simply picks the sender of the first response received.

3.2 Simplified pull mechanism

In the mainstream decentralized P2P systems, a node wants to get a content of its interest would flood its request and awaits for positive responses from the suppliers, then makes the connections to the suppliers to start content retrieval after electing a set of suppliers; this process repeats until the maximum downstream rate is reached or the content retrieval is completed. This pull-based approach works out okay in physically connected networks as the connections are highly reliable, since the requesting node almost always can successfully establish its connections back to the responding suppliers.

In a highly dynamic wireless environment, where nodes move around and attach and detach from the network randomly, this traditional pull-based query technique will not work for the simple reason that the requesting node is not taking advantage of a short live route while waiting for a collection of positive response from servers . ASAP reuses the concept of flooding the request to reach out to the suppliers in the network, but instead of wait for multiple servers, the requester will immediately confirm the retrieval with the very first response received from any supplying node, any further responses from different supplier will be simply ignored. According to a study conducted by Moraes et al[7], using numerous number of short-lived TCP connections help to reduce negative mobility impacts on streaming sessions; and that UDP enable a smoother WMN (Wireless Mesh Network) adaptation for streaming technologies. Choosing UDP over TCP would

enable the P2P system to utilize the supplier even in the case of a supplier departing from the network shortly after starting serving.

3.3 Partitioning delivery paths with relaying nodes

Based from the relaying concepts used in MANET and P2P systems, ASAP also make use of relaying nodes in a similar way but with a twist. Unlike the relaying node's responsibility in most MANET and P2P systems, that its role is to transparently help with traffic propagation to the next hop or node; relaying nodes in ASAP turns itself into a requesting node on behave of the "original" requesting node (the source node of the received request). In other words, from the view of the "original" requesting node, the relaying node is the supplier, and from the view of the "original" supplier, the relaying node is the requester, as depicted in Figure 1. The idea is to store a list of the "original" requester IP at relaying nodes, and some internal state regarding to the continuous block of segments the "original" requester is interested in retrieving.

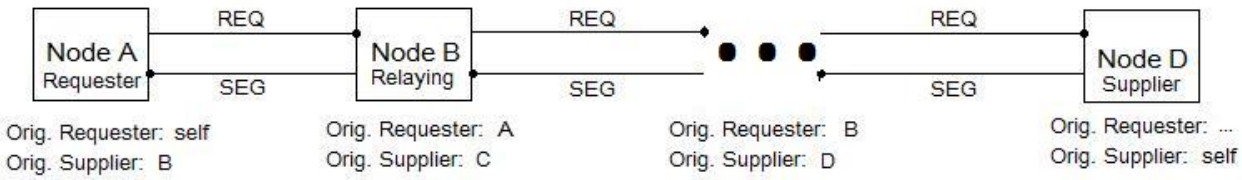

Figure 2. Role of a relaying node

By making the relaying node act like a real requester and supplier, a delivery path that involves multiple relaying nodes could essentially be broken down into multiple independent requester-to-supplier paths, which ultimately allows faster recovery from link failures at the relaying node, with a buffer that stores relaying traffic temporarily with a timeout; de-multiplexing equivalent requests from multiple "original" requesters; eliminating duplicate content segments

from multiple “original” suppliers. Performing these tasks at the relaying node will greatly reduce the impact of link failure, and the amount of wasted bandwidth by redundant traffic.

3.4 Fast route changes detection

In VANET, routes are consistently changing due to the high mobility of nodes; this has a couple of undesired side effects: 1. the underlying ad-hoc routing protocol introduces more control traffic to refresh its routing table with very small update intervals, or its routing table cannot keep up with the frequent changes with longer update interval; 2. data transmission might end up taking suboptimal physical routes as the logical routes built with intermediate nodes could have changed to involve more hops in a very short amount of time.

To neglect most of these side effects, ASAP sends all traffic with TTL of 1, this means that all traffic can only be successfully sent to direct neighbors within transmission range; as there is no retransmission in UDP, when each time a node received an ASAP message (control or data), it starts a timer with a dynamically computed timeout T using equation (1); when this timer expires before receiving a new message from the peer, the peer is marked as disconnected.

$$T = 1 / N * \beta \quad (1)$$

where N is the number of segments needed in one second of playback, and the β multiplier is used to add buffer to allow transmission delay.

With this simple modification of TTL and the use of timeout, ASAP is highly decoupled from the routing layer; route tables only needs to be up-to-date with direct neighbors, meaning routing layer can use longer update interval; and when the timer expires at the end client node, it will broadcast a new request to rebuild the logical route to a serving peer. However, when timers

expired at relaying nodes, no attempt to rebuild logical route will be made to avoid excessive amount of broadcast messages.

3.5 Sliding playback buffer as receiver window

Most P2P systems are designed for file sharing, which make sense to strike for maximize downstream data rate so the clients can complete the download as soon as possible. Common P2P systems achieve high downstream data rate by parallel content retrievals; the interested content is split into numerous small chunks of data segments, each of the segment then is treated as an individual data block/segment that could be retrieved independently. In addition, P2P file sharing commonly make use of some prioritization scheme to decide which segment to be retrieved first, i.e., rarest first, such scheme allows the P2P network replicate content availability throughout the course of the peers' stay in the sharing community. A high availability equates to bigger set of suppliers and therefore requester can achieve high downstream rate by performing more parallel content retrieval. Due to the unique characteristics of media streaming, this parallel scheme is not suitable. The perceived user experience of media streaming solely depends on the quality and smoothness of the playback, regardless of the time it took to complete the retrieval. In contrast to the P2P file sharing systems, P2P streaming is better off being conservative with its bandwidth consumption, especially in wireless environment where the transmission medium is shared by all users within the transmission range; being too aggressive in bandwidth consumption will only lead to congestion, and ultimately slow down the network.

ASAP only strike to offer playback smoothness by retrieving enough segments to keep its playback buffer full as much as possible, and limit the system from delivering any segment beyond the playback buffer boundary. Client requests includes the identifiers of the first and last segment that defines the boundary of the buffer, so senders know to stop when it reaches the end of the

buffer. When the client finish filling the buffer, the playback buffer boundary will be pushed forward , and then the client will broadcast a request to fill the next buffer for more continuous segment retrieval.

3.6 Optional reuse of existing delivery path

ASAP implements an optional feature that allow it to reuse an existing delivery path when a client advances its playback buffer. Instead of tearing down the existing route and broadcast a request to rebuild a route, the end client will send a sync message back to its immediate serving node, this sync message contains the new boundaries of the buffer; the immediate serving node then updates its internal state about the buffer boundaries, then keeping forward this sync message until it hits the end serving node. When the end serving node receives this sync message, it will send back the first segment as indicate by the boundaries. This optional feature is experimental to see the effects on reusing existing route, which will be discussed as part of the analysis of the simulation results.

3.7 Message format

All traffic sent in ASAP are following a particular format, as depicted in Figure 3 below, which consist of two parts, the meta header, and the data portion; all control message contains only a meta header without the data portion, which is a segment by itself and would only be embedded in data message.

Meta Header						
0	1	5	7	9	13	17 21
Type	Stream ID	Segment ID	End Segment ID	Sender	Origin	Message ID
Data (segment)						

Figure 3. ASAP message format

"Type" is a 1 byte field to specify the type of the message, and it is mandatory for all kinds of ASAP messages; the possible values are - REQ, DATA, ACK_SYNC, ACK_CANDIDATE, and ACK_CONFIRM, represents request, data, sync, candidate, and confirm messages.

"Stream ID" uniquely identifies the video stream with field length of 4 bytes, this field is also mandatory for all messages.

"Segment ID" is a 2-byte value that uniquely identifies a segment in a video stream, which is another mandatory field.

"End Segment ID" is the segment id that indicates the last segment of the continuous block of segment the being requested, this 2-byte field is only included in request and sync messages.

"Sender" field contains the 32-bit integer representation of the IP address of the node which is sending the message, if support of IPv6 is desired, this field needs to include the address family and 128-bit for the address; this field is mandatory for all messages

"Origin" field is 4-bytes long and specifies the IP address of the initiator of a request; this field is set by a client node and is only required for request messages; relay nodes will copy this field for any request it forwards.

"Message ID" is a 32-bit unsigned integer used to uniquely identifies a control (request, sync, and confirm) message sent from client / relay nodes for a particular stream, the initial values starts at 0, and increment for every control message sent; this field is not included in data message. This field is stored by relay and server nodes, and it is used to ignore any control messages that might be delivered out of order or lingered around in the network.

3.8 Client node

A client node is the only kind of node in ASAP that initiates a request, by broadcasting a request message via UDP, as long as either of the two following events occurred:

1. The request broadcast timed out and there is not yet an active supplier

2. The current active supplier became inactive

Initially, the client node will first determine the continuous block of segment to retrieve for a streaming session. The first continuous block starts with the first segment of the stream, with ID equals to 1, and ends with the segment with the ID equals to 1 + *N*, where *N* is the numbers of segment needed for filling the playback buffer, calculated with equation (2):

$$N = B * T / S \quad (2)$$

Where B is the bit rate of the stream (in bits per second), T is the number of seconds of playback data to buffer, specified by the media player, S is the size of a segment (in bits)

When the block is determined, the client will construct a request using the start segment and the end segment of the block as the value for "Segment ID" field and "End Segment ID" field in the meta header, then timer to wait for a supplier candidate; in case of multiple candidates responded, the client node would only respond to the first candidate message it received. When retrieval of a block completed, the client will move to the next block; when the optional reuse of existing delivery path feature (section 3.6) is turned off, the client will simply mark the supplier as inactive and erase any internal state related to it, and broadcast a new request for the next block; on the other hand, in attempt to reuse the existing delivery path, the client will send a sync

message back to the supplier, requesting for the next block. This process repeats until the whole stream is retrieved.

Due to the unstable nature of VANET, packets might get lost, and packets might be delivered out of order via UDP. When a delivery path is broken while retrieving a block, the partially retrieved block will contain holes in it instead of a nice continuous sub-block. Therefore, client node needs to make subsequence requests to fill these holes when a broken route occurred and the supplier detected inactive.

Whenever a client node receives a data message, it will send out a sync message to retrieve the next missing segment. In order to not waste bandwidth to retrieve any segment that already passed behind the current playback position, client nodes keep track of the current playback position as in segment ID, and ignore any missing segment that has an ID that is smaller than the ID of the current playing segment.

3.9 Server node

A server node in ASAP is the node that contains the whole continuous block being requested, also known as a seed in P2P systems. When a server node receives a new request, and the IP address of the client node (from "Origin" field), the message ID (from "Message ID" field). For any request a server node receives, it checks if it has already responded to this request or a newer request from the same client, and discard older or same requests that initiate by the same client, as the server node will receive multiple copies of the same request coming from different relay nodes.

When a server node responds to the requester with a candidate message, it will use the IP address taken from "Sender" field, to send to the relay node, in cases where there is no relay node in the delivery path, the requester is the same as the client node, therefore the "Sender" field and

"Origin" field contains the same IP address. Then the server node will wait for a confirm message as an indication that the client node committed to retrieve from this server node, the "Sender" field of the confirm message will be stored as a receiving node, so it can ignore any further requests coming from the same node; The "End Segment ID" field is also stored at this stage, so when the server sends out the last segment in the continuous block, it will clear out all internal states related to the receiving node so it can accept request from the same node again; the server node will also clear out internal states about a receiving node when client is detected inactive, as the timer (described in section 3.4) expires.

3.10 Relay node

Relay node is the most complex and important role in ASAP; in a nutshell, it is a specialized proxy that not only passes traffic between client and server nodes, but also multiplexes requests upstream and demultiplexes data downstream. This special role is the main strength in ASAP.

When a node receives a request message, and determined that itself does not have the continuous block to serve, it will take on the relaying role as long as the any of the following conditions is met:

1. it is not currently relaying for the requested stream

2. requested block is within the current relaying block

3. requested block intersects with the tail of the current relaying block

4. start of the requested block is adjacent to the end of the current relaying block.

When a node decides to relay, it will adjust the relaying block information accordingly if necessary, and attaches the requester to the relaying block; then the relay node will broadcast a

request for the relaying block if it does not have a supplier yet, and start a timer to expire the relay event if it does not receive a candidate message within the specified time, as the server node might respond to a different delivery path that does not involve this relay node, or the candidate message just go lost during transmission; relay node will detach the requester from the relay block when its relay event expires. If the relay node already has a supplier serving the relaying block, it will not broadcast a request and respond to the requester with a candidate message.

When a relay node receives a candidate message for a relaying block, it will ignore the candidate if the relay block has already assigned a supplier via an earlier candidate message, otherwise it will assigned the supplier and forwards a candidate message to all requesters attached to the relaying block, and waits for a confirm message before committing retrieval.

When a relay node receives a confirm message, it will process it like a server node (as described in section 4.2), and then forward a confirm message upstream to its supplier if it forwarded a request via broadcast and this is the first time it received a confirm message for the relaying block. The requester that sent the confirm message is marked as a receiving node at this stage, therefore any further requests from this node will be discarded. The relay node will not forward the confirm message upstream for any further confirm messages sent from attached requesters, it will simply marked it as a confirmed receiver so it can ignore future requests from confirmed receivers.

As soon as a relay node is committed retrieving a relay block, the rest of the retrieval is independent of the downstream requesters, the relay node itself will continue to retrieve the relaying block independently, as long as there is one or more active downstream requester attached to the relaying block. If all downstream requesters became inactive, it will stop retrieval

and clears out all internal states related to the relaying event, but always keeps the data it has already retrieved.

It is possible that a downstream requester send a sync message to retrieve a segment that is not yet received from a upstream supplier at a relaying node, therefore, in such case the relaying node will register a callback to be notified when the requested segment becomes available, and then sends it to the downstream requester.

3.11 Sessions and multiple roles

It's important to note that a node can play multiple role at anytime such that any node can be a client, server, and relay node at the same time; and all roles and internal states discussed in this chapter are session independent, each media stream has its own session.

Chapter 4. Performance evaluations

4.1 Environment and parameters

To validate the performance of ASAP, a vanilla P2P protocol is implemented and used to compare and contrast with ASAP and E-ASAP. The main differences between the three implementations are described in Table 1 below.

	Vanilla	ASAP	E-ASAP
Transport layer	UDP for request broadcast TCP for everything else	UDP	UDP
Parallel retrieval	Yes	No	No
Application layer relay	No	Yes	Yes
Reuse delivery path	Not applicable	No	Yes

Table 1. Vanilla, ASAP, and E-ASAP implementation differences

Metrics used includes : 1. packet loss ratio, to analyze transmission efficiency; 2. duplicate send / receive ratio, to analyze the waste of sending same data to end clients; 3. average hop count of delivering data, to analyze the usage of optimal routes; 4. the ratio of missed playback, to analyze the timeliness of transmission.

For the purpose of a comprehensive view and a fair comparison between the two protocols, we run the simulation under the same environment as following:

- 1200 meters x 1200 meters moving area
- 30 vehicles, with each vehicle moves at a randomly chosen speed between 40 to 80 mph, at random directions

- wireless media of 802.11a standard, 300 meters transmission range, and 54Mbps bandwidth
- video stream with bit rate of 256 Kbps, duration of 300 seconds
- each video segment contains 1 second of playback
- playback buffer of 15 seconds
- β in equation (1) uses value of 0.8 to create buffer for transmission delays
- 2 possible servers, one locate at edge of the cluster, hard to reach by other nodes, the other locate at the center of the cluster, easy to reach by other nodes

The above environment setup is used for all variations of:

- 1 server node with 2 / 5 / 10 client nodes, rest being relay nodes.
- 2 server nodes with 2 / 5 / 10 client nodes, rest being relay nodes.

For the 1 server variations, the hard to reach server is used.

4.2 Results

4.2.1 Mixture of 40-80 mph movement speed

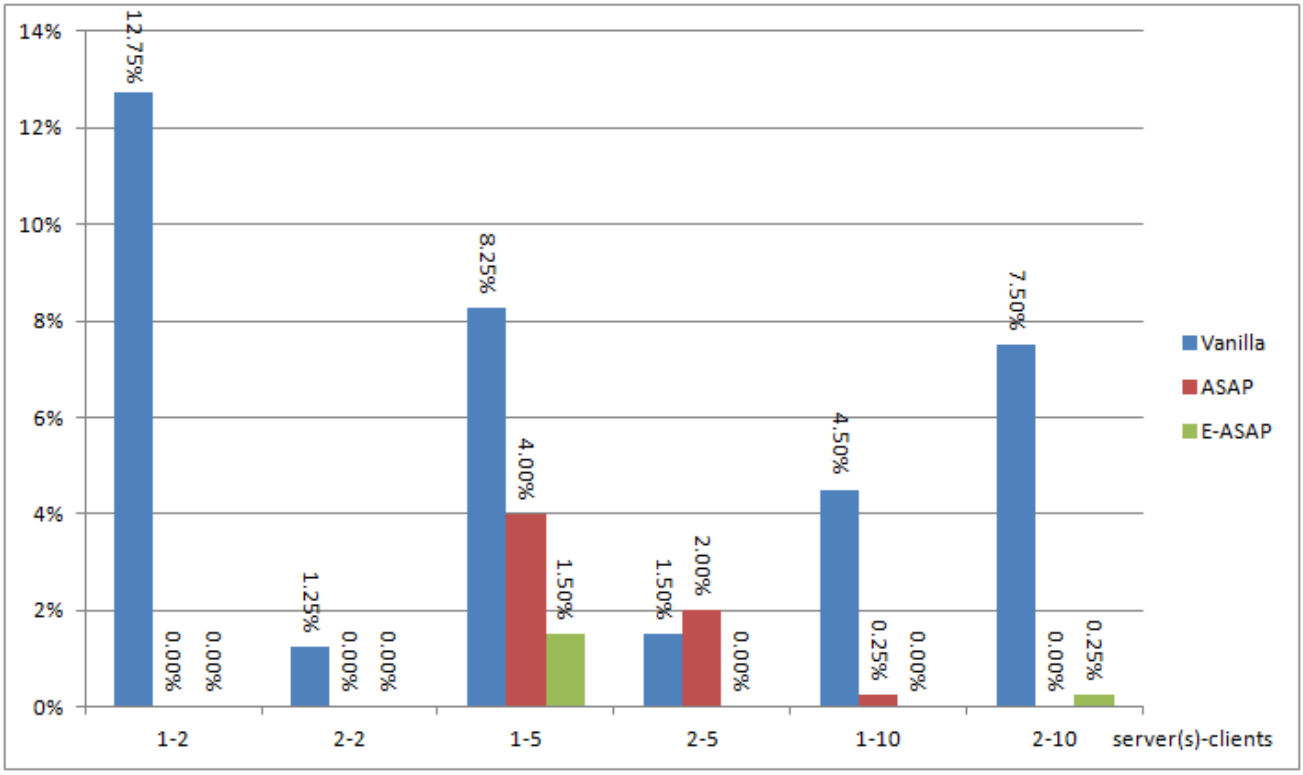

Figure 4. Missed playback (40-80 mph)

Both ASAP and E-ASAP improve the smoothness of playback by reducing missed playback. As indicated by the results. Although ASAP has 0.5% more missed playback in the 2 server 5 clients case, E-ASAP still achieved 0%. Since the easy to reach server has high availability to the clients, ASAP is at disadvantage with its broadcast messages in attempt to rebuild better router. Note how Vanilla has more missed playback in the 2 servers 10 clients compare to the 1 server 10 clients setup, this is because the performance of P2P systems is only depending on utilizing the upload capability of peers, not the number of peers, in this case, Vanilla fail to utilize the extra server.

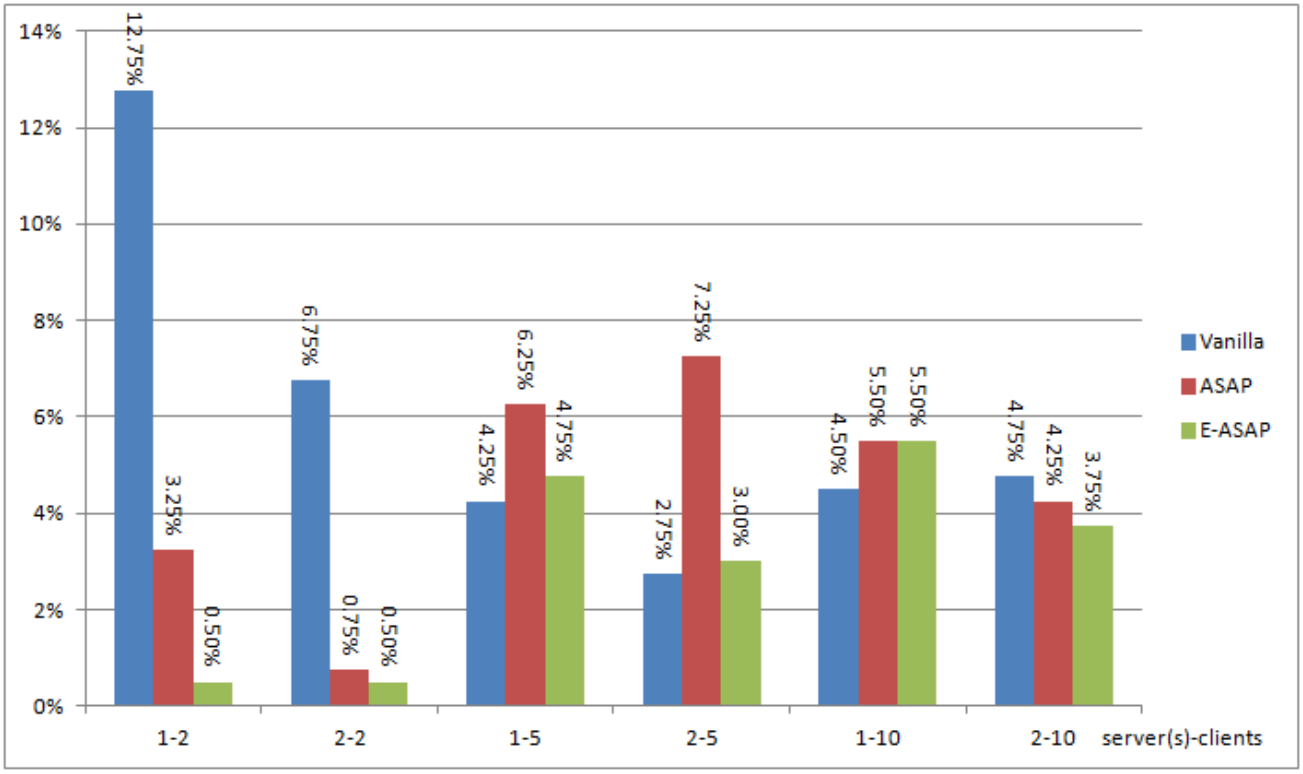

Figure 5. Receive duplicates (40-80 mph)

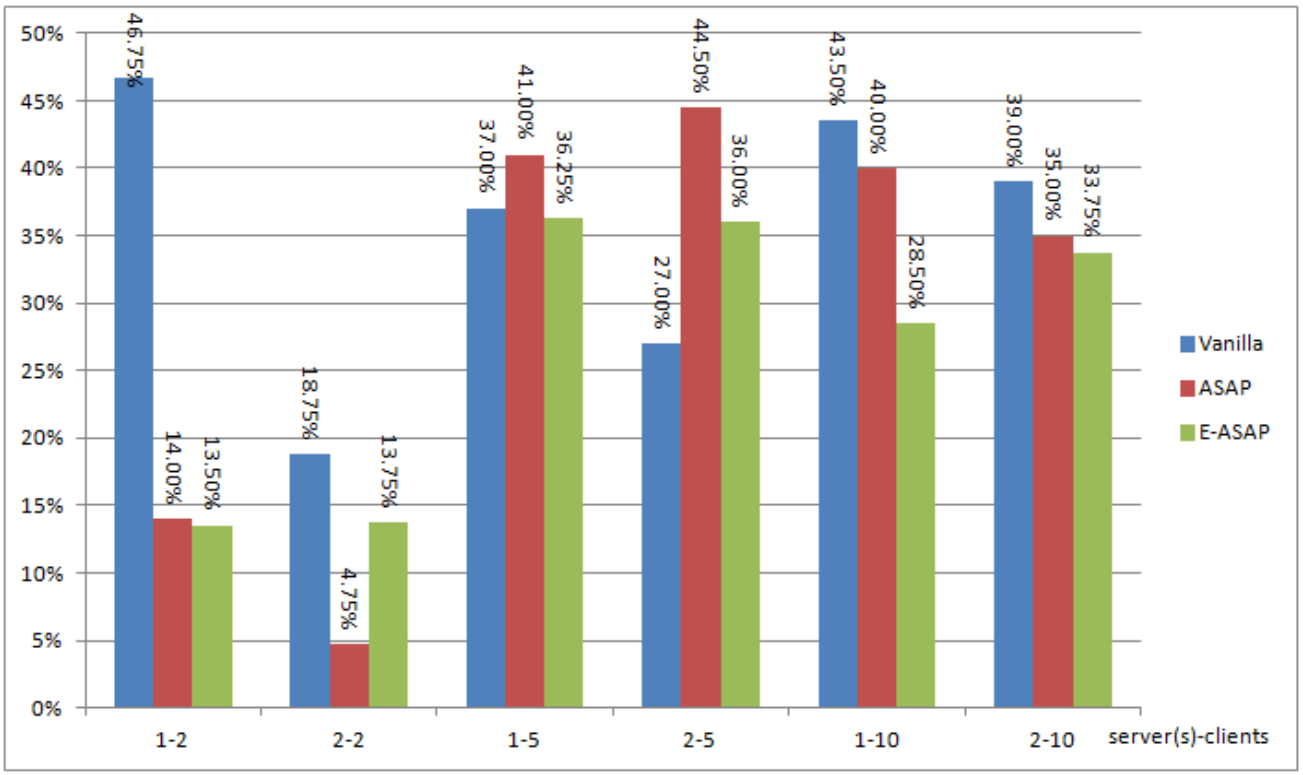

Figure 6 . Send duplicates (40-80 mph)

Duplicate send ratio in Vanilla is higher than ASAP and E-ASAP, except the 5 clients scenarios; however, since we are not counting TCP retransmissions as part of duplicate sends, and Vanilla has more missed playback implying there is less sends from the server; vanilla P2P results could have shown false positive in this area. Since ASAP has more broadcast requests than E-ASAP without reusing delivery path, it is expected to have higher duplicated sends than E-ASAP caused by higher packet loss possibility due to traffic collision.

4.2.1 Various speed breakdown

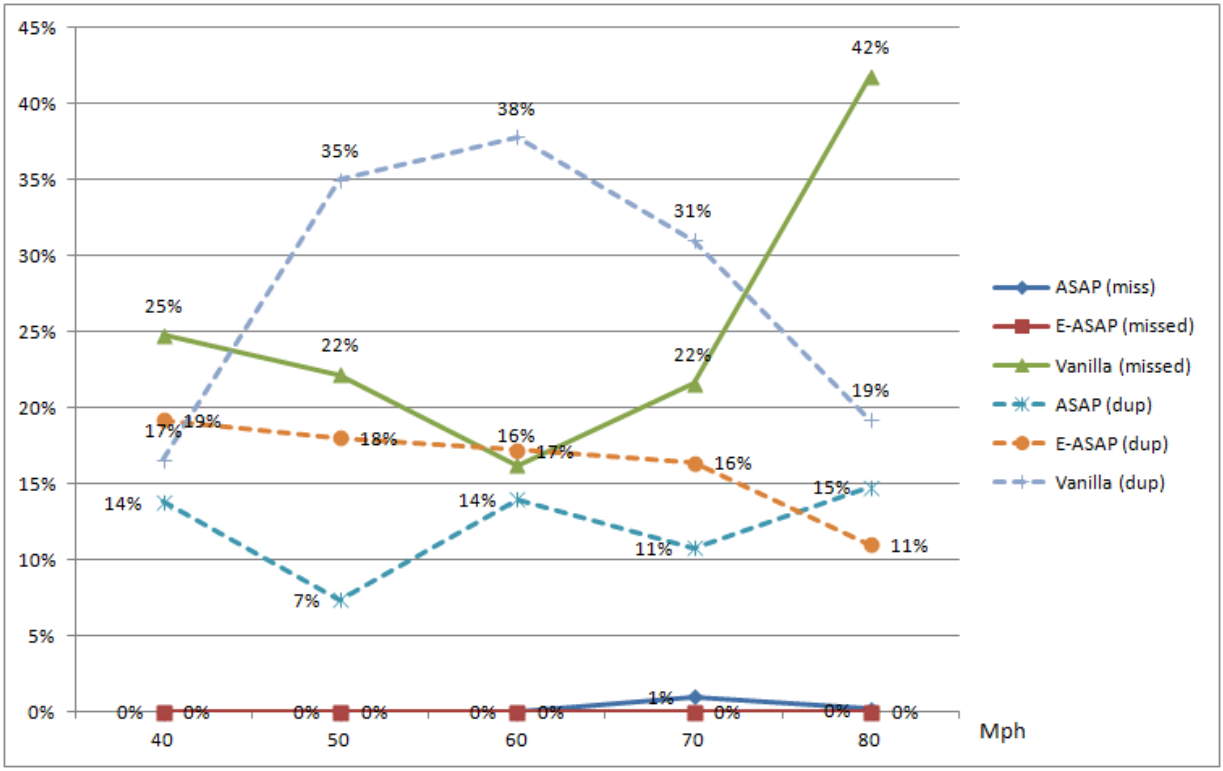

Figure 7. 1 server 2 clients

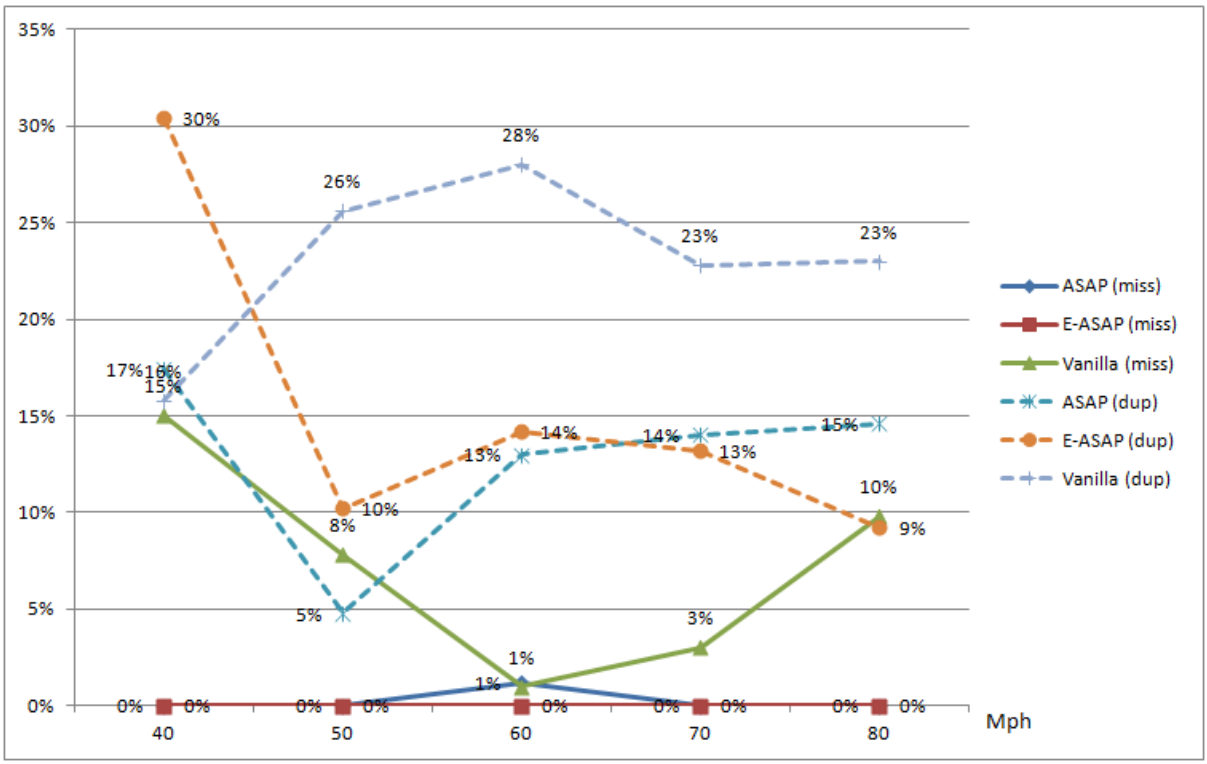

Figure 8. 2 servers 2 clients

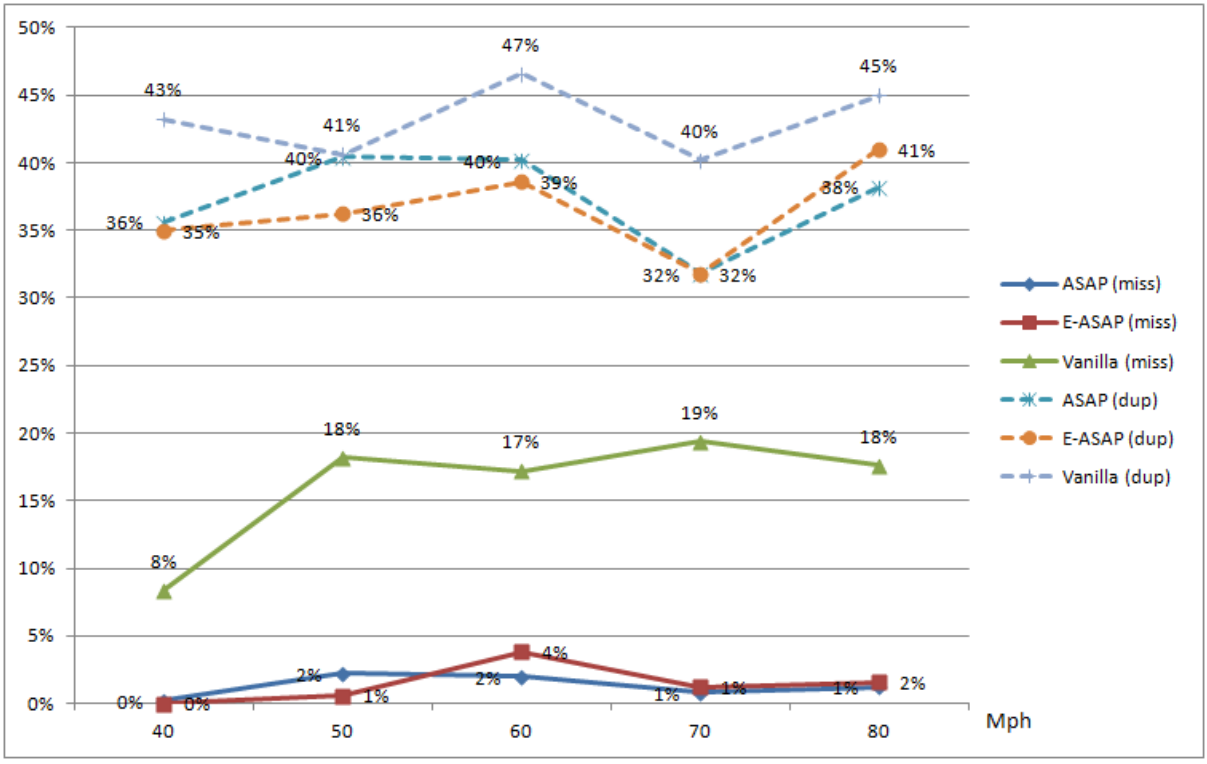

Figure 9. 1 server 5 clients

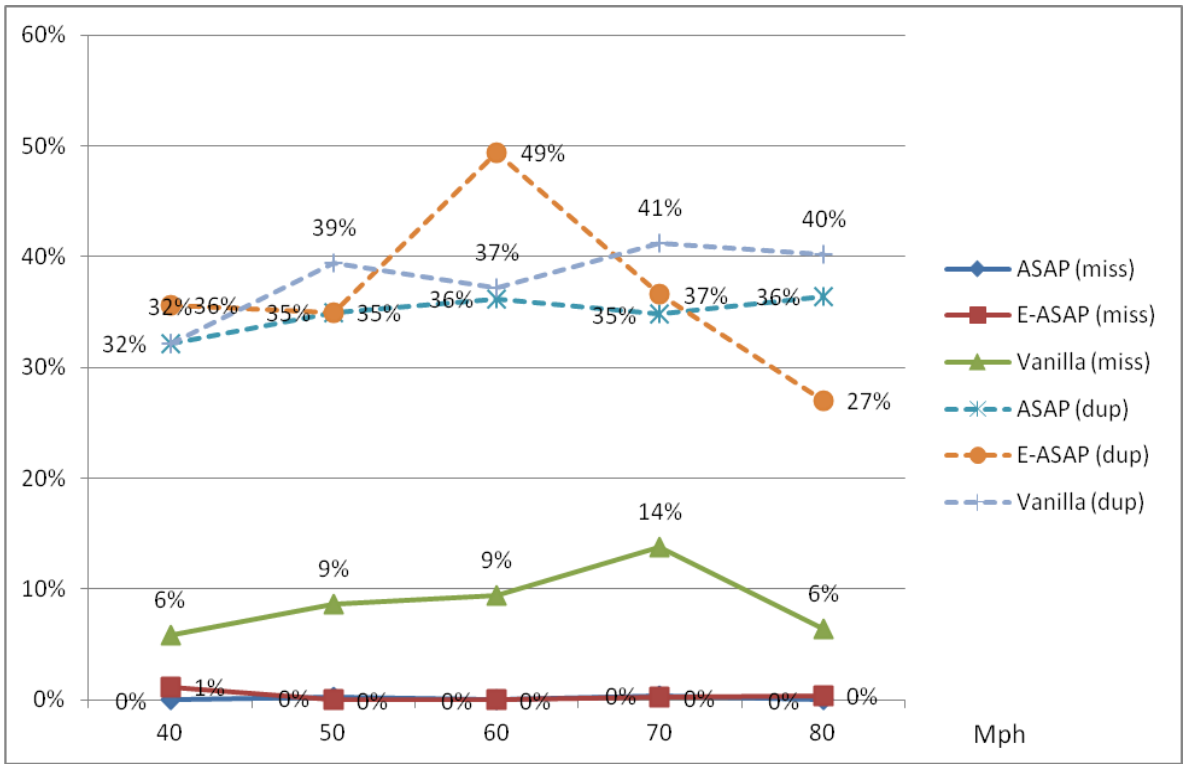

Figure 10. 2 servers 5 clients

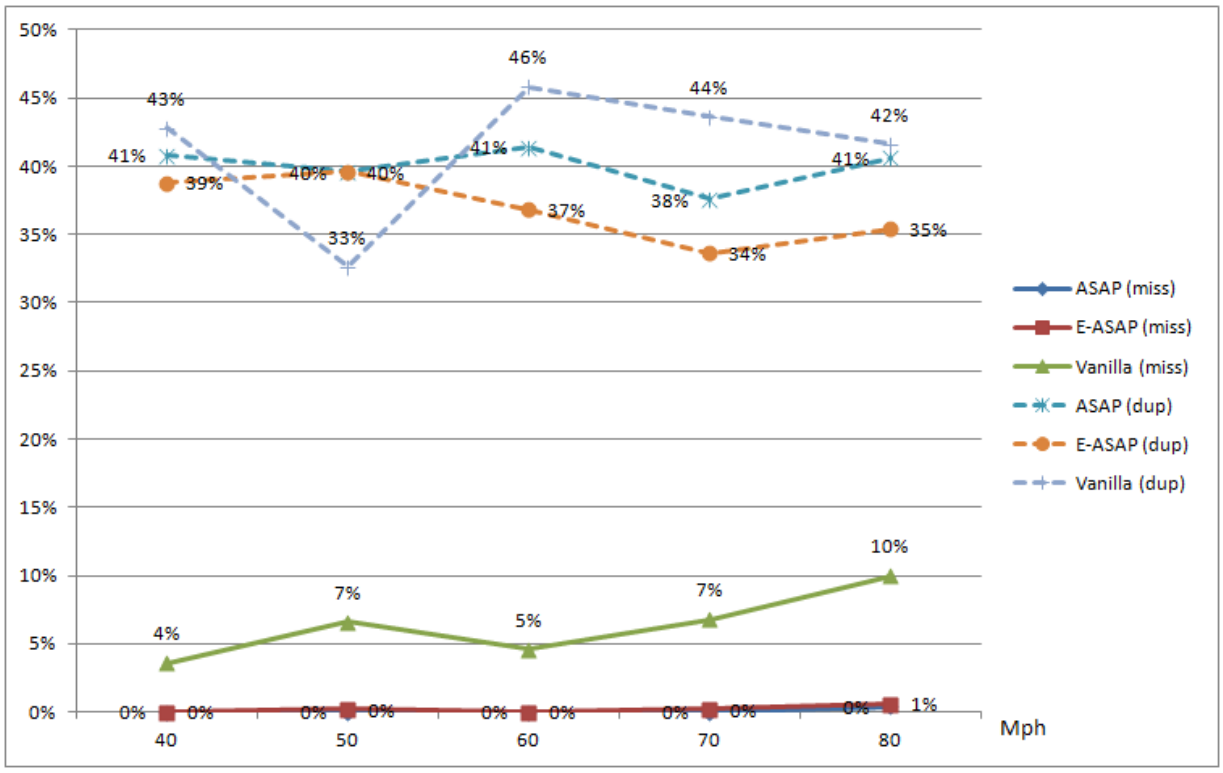

Figure 11. 1 server 10 clients

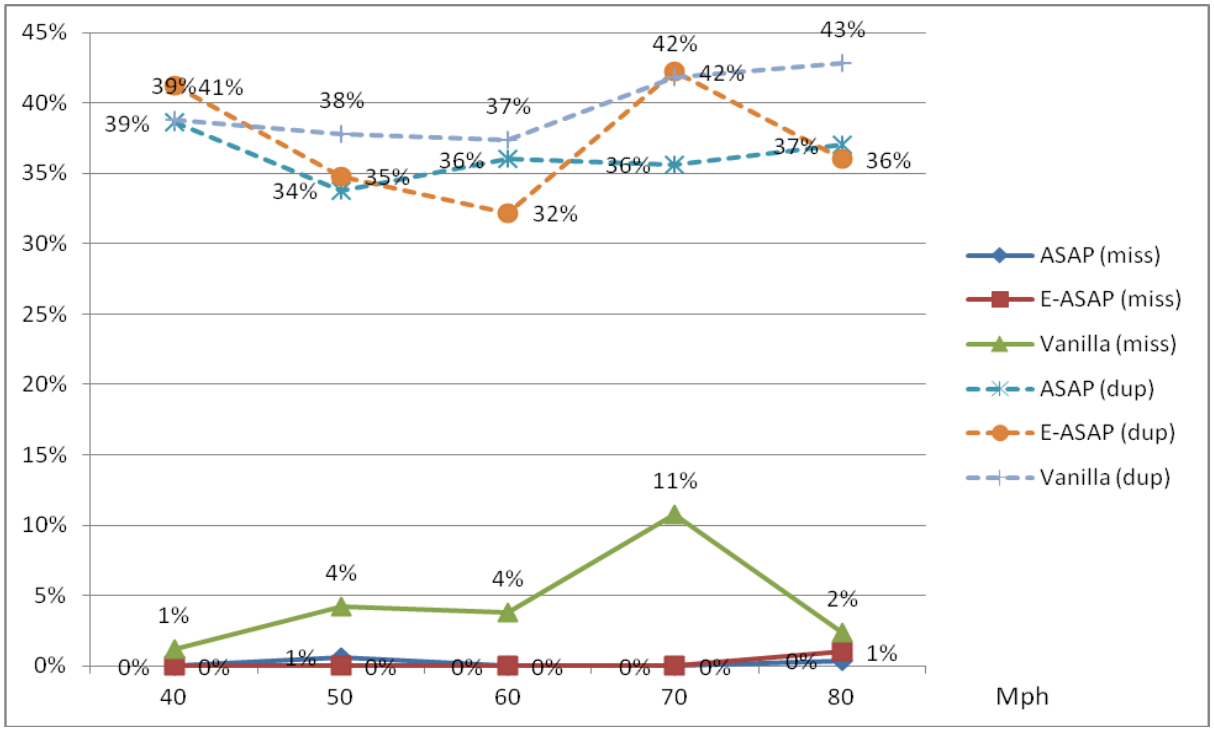

Figure 12. 2 servers 10 clients

In general, vanilla performance degrades as the movement speed increases, and it is more obvious with the 1 server cases and the missing easy to reach server makes the Vanilla systems more prone to route failures. On the other hand, ASAP and E-ASAP is able to perform consistently well across the movement speed range. It is also obverse that ASAP and E-ASAP reduced the average hop counts traversed for all delivered data, this is expected behavior as the caching ability in ASAP and E-ASAP provides content availability at a closer distance to all clients.

.

Chapter 5. Conclusion

In this paper, we seek the possibly of video streaming over VANET as an alternative to using cellular data network, so to reduce the burden on mobile service providers. Decentralized P2P is the suitable technology to use for content delivery in ad-hoc wireless network, because of its nature of flexibility with network topology and scalability that can grow supplying bandwidth as the number of nodes grows. However, traditional P2P content sharing systems are mostly TCP oriented and rely heavily on stable connections, which is exactly what extremely mobile environments like VANET lacks; recent studies have also shown that the most influential factors in P2P system performance is maximizing utilization of upload capability provided by peers; TCP based P2P systems in mobile environments suffers from route changes; P2P systems inherent a inefficiency in delivery the same content multiple times.

Various recent proposals of enhancement to improve P2P streaming by using a cache entity to help reducing duplicate content delivery; and a number of proposals have suggested different approaches to adaptively choose an optimal route or peer for content retrieval in a constantly changing network topology. These studies have shown proven benefits an adaptive protocol that can embrace the frequent route changes, and reduced duplicate content delivery boost P2P performance. Based on these findings, we present ASAP as a simple application layer solution for ad-hoc video streaming in VANET, with goals to have an adaptive protocol, thus UDP is chosen and a fast route change detection built on top it, and introducing a specialized relay node in the P2P overlay to cache content, multiplexing client requests, and demultiplexing data delivery.

Our experiment results and analysis in the last chapter confirmed the strengths of ASAP over a vanilla implementation of traditional approach of P2P streaming; ASAP improves the smoothness of the playback by delivering content more efficiently and timely.

Yet, there are possible enhancements to ASAP that are not included in this study due to the limit of time. ASAP could definitely uses a cache replacement algorithm to cap its storage footprint in storage usage, as the current design does not include one to remove cached content. ASAP current avoid retrieve any segment that already passed its playback deadline, this is not ideal as any segment that soon to be played might also miss its playback deadline by the time client node receives it, this could be further enhancement by using a dynamically calculated offset factoring in the current retrieval rate.

ASAP might also benefit from a more sophisticated dynamic scheme for reusing existing delivery path; our results has shown that reusing the existing delivery path could have improve performance in some cases but worsen performance in other scenarios, so a dynamic scheme to determine the likelihood of finding a better route can help make the decision of whether or not to try reuse the route; client node can keep track of transmission delays for the first segment and the last segment received, take the difference and only rebuild the delivery path if the difference indicates a degraded route; or some geographical data can be used instead of transmission delay.

Appendix

Software Development

NS-3 (http://www.nsnam.org/) is a C++ network simulation framework used for implementing the software for our simulations. NS-3 provides a solid library of the lower layers (layer 1 to 4) of the network stack, so the implementation mainly focused on the higher layers including session, presentation, and application. NetAnim is an external animator framework that can be integrated with NS-3 (http://www.nsnam.org/wiki/index.php/NetAnim), which is used in this project to visualize the simulation in cases where we need to observe movement patterns of peers and route changes, so that proper roles of peers can be selected.

ASAP in its initial proposed design uses a push model that is later in simulations confirmed to be not suitable for VANET. Since ASAP adopt agile development practices such that it used abstraction design by following OO principles and using interfaces; we were able to make rapid changes to our software, observe and confirm design flaws and improvements. ASAP was evolved into its current after numerous trails and errors in small implementation increments.

References:

1. Chan, Hung Nguyen; Van, K.N.; Hoang , Giang Ngo; “Characterizing Chord, Kelips and Tapestry algorithms in P2P streaming applications over wireless network” Communications and Electronics, 2008. ICCE 2008. Second International Conference on 4-6 June 2008 Page(s):126 – 131
http://ieeexplore.ieee.org.libaccess.sjlibrary.org/stamp/stamp.jsp?tp=&arnumber=4578945&isnumber=4578921

2. Chen, Jun; Du, Xu; Cheng, Wengqing; Xu, Jing. "Providing Efficient P2P VoD Service with Priority Based Scheduling". Wireless Communications Networking and Mobile Computing (WiCOM), 2010 6th International Conference, Sept 2010. page(s): 1

3. Dai, Dan; Cao, Lin; Zhou, Xuemei; Zhao, Yiwei; “Cache mechanism in P2P streaming media system”, Networking and Digital Society (ICNDS), 2010 2nd International Conference on 30-31 May 2010, Page(s): 376

4. Gurses, E.; Kim, A.N.; “Maximum Utility Peer Selection for P2P Streaming in Wireless Ad Hoc Networks” Global Telecommunications Conference, 2008. IEEE GLOBECOM 2008. IEEE Nov. 30 2008-Dec. 4 2008 Page(s):1 – 5
http://ieeexplore.ieee.org.libaccess.sjlibrary.org/stamp/stamp.jsp?tp=&arnumber=4698124&isnumber=4697775

5. He, Yifeng; Guan, Ling. "Solving Streaming Capacity Problems in P2P VoD Systems". Circuits and Systems for Video Technology, IEEE Transactions, Vol 20, Issue:11, Nov. 2010. Page(s): 1638.

6. Jin, Xin; Kwok, Yu-Kwong. "Cloud Assisted P2P Media Streaming for Bandwidth Constrained Mobile Subscribers". Parallel and Distributed Systems (ICPADS), 2010 IEEE 16th International Conference, Dec. 2010. page(s): 800

7. Lai, Chung-Yi; Hsu, Chiun-Sheng; Shang, Tzyh-Jong; “Improved P2P Streaming in Wireless Networks Utilizing Access Point P2P Agents” Multimedia Workshops, 2007. ISMW '07. Ninth IEEE International Symposium on 10-12 Dec. 2007 Page(s):58 – 62
http://ieeexplore.ieee.org.libaccess.sjlibrary.org/stamp/stamp.jsp?tp=&arnumber=4475944&isnumber=4475929

8. Li, Ying; Li, Zhu; Chiang, Mung; Calderbank, A.R.; “Energy-Efficient Video Transmission Scheduling for Wireless Peer-to-Peer Live Streaming” Consumer Communications and Networking Conference, 2009. CCNC 2009. 6th IEEE 10-13 Jan. 2009 Page(s):1 – 5
http://ieeexplore.ieee.org.libaccess.sjlibrary.org/stamp/stamp.jsp?tp=&arnumber=4784766&isnumber=4784684

9. Liu, Chia-Yi; Wang, Kuochen; Hsieh, Yi-Ling; “Efficient push-pull based P2P multi-streaming using application level multicast”, Personal Indoor and Mobile Radio Communications (PIMRC), 2010 IEEE 21st International Symposium on 26-30 Sept. 2010, Page(s): 2586

10. Liu, Pin-Chuan; Chiang, Tsun-Chieh; Chien, Yi-Fan; Shih, Wei-Kuan; Hu, Chih-Lin; “Improving Mobile Peer-to-Peer Streaming Service with BitTorrent-Like Redundant Tracker” Mobile Data Management: Systems, Services and Middleware, 2009. MDM '09. Tenth International Conference on 18-20 May 2009 Page(s):643 – 648
http://ieeexplore.ieee.org.libaccess.sjlibrary.org/stamp/stamp.jsp?tp=&arnumber=5089017&isnumber=5088899

11. Luo, Haiyan; Ci, Song; Wu, Dalei; “A Cross-layer Optimized Distributed Scheduling Algorithm for Peer-to-Peer Video Streaming over Multi-hop Wireless Mesh Networks” Sensor, Mesh and Ad Hoc Communications

and Networks, 2009. SECON '09. 6th Annual IEEE Communications Society Conference on 22-26 June 2009 Page(s):1 – 9
http://ieeexplore.ieee.org.libaccess.sjlibrary.org/stamp/stamp.jsp?tp=&arnumber=5168902&isnumber=5168895

12. Moraes, I.M.; Campista, M.E.M.; Costa, L.H.M.K.; Duarte, O.C.M.B.; Duarte, J.L.; Passos, D.G.; de Albuquerque, C.V.N.; Rubinstein, M.G.; "On the impact of user mobility on peer-to-peer video streaming" Wireless Communications, IEEE Volume 15, Issue 6, December 2008 Page(s):54 – 62
http://ieeexplore.ieee.org.libaccess.sjlibrary.org/stamp/stamp.jsp?tp=&arnumber=4749748&isnumber=4749736

13. Peng, Hui; Shuang, Kai; Han, XiaoYong; Lin, XiuQin. "Research on the Mesh-based P2P streaming network scale". Communications and Mobile Computing (CMC), 2010 International Conference, April 2010. page(s): 324

14. Rodrigo, Javier; Leal, Raquel; Martin Encarna. "Peer-to-peer IPTV service Impact on Network Traffic", Digital Elecommunications (DCDT), 2010 Fifth International Conference. Page(s): 132-137

15. Salta, Nuno; Morla, Ricardo; Ricardo, Manuel; "Improving P2P video streaming in wireless mesh networks", Ad Hock Networking Workshop (Med-Hoc-Net), 2010 The 9th IFIP Annual Mediterranean, 23-25 June 2010, Page(s): 1

16. Tan, Enhua; Guo, Lei; Chen, Songqing; Zhang, Xiaodong; "SCAP: Smart Caching inWireless Access Points to Improve P2P Streaming" Distributed Computing Systems, 2007. ICDCS '07. 27th International Conference on 25-27 June 2007 Page(s):61 – 61
http://ieeexplore.ieee.org.libaccess.sjlibrary.org/stamp/stamp.jsp?tp=&arnumber=4268214&isnumber=4268148

17. Wang, Mea; Li, Baochun; "Network Coding in Live Peer-to-Peer Streaming" Multimedia, IEEE Transactions on Volume 9, Issue 8, Dec. 2007 Page(s):1554 – 1567
http://ieeexplore.ieee.org.libaccess.sjlibrary.org/stamp/stamp.jsp?tp=&arnumber=4378432&isnumber=4378421

18. Zhang, Yubao; Wang, Hui; Li, Pei; Jiang, Zhihong; Gao Chao. "How can Peers Assist each other in Large-scale P2P-VoD Systems". Multimedia Information Networking and Security (MINES), 2010 International Conference, Nov. 20101. page(s): 198

19. Zhu, Yingnan; Zeng, Wenjun; Liu, Hang; Guo, Yang; Mathur, S.; "Supporting Video Streaming Services in Infrastructure Wireless Mesh Networks: Architecture and Protocols" Communications, 2008. ICC '08. IEEE International Conference on 19-23 May 2008 Page(s):1850 – 1855
http://ieeexplore.ieee.org.libaccess.sjlibrary.org/stamp/stamp.jsp?tp=&arnumber=4533391&isnumber=4533036