P2P communication efficiency algorithms

Milea Robert Roxana Irimia Teodora Hoamea

April 2022

Contents

1 Introduction

Because of the evolution of technology the use of available bandwidth and computing power for videoconferencing, streaming or file sharing is a daily basis situation. Because of that we need efficient algorithms which offers quality and speed to the final user.

We started by analyzing Gnutella, one of the first real word application of a gossip algorithm in P2P network communication, followed by a comparison with the torrent model. We will present the key factors that made the Gnutella protocol inefficient in file sharing, why it's model is used in blockchain and what made torrents so popular.

We continued by presenting GONet, an algorithm based on gossip protocols that tries to solve the problem of redundancy of gossip-based protocols and also has a better continuity and end-to-end delay than a tree based overlay.

For audio/video streaming on mobile devices, José-Vicente Aguirre, Rafael Álvarez, Leandro Tortosa, and José-Francisco Vicent in their paper "P2P Audio/Video Protocol with Global Positioning Data in Real Time for Mobile Devices" developed an algorithm that is using encoding of global positioning coordinates to transmit securely the location of the audio/video stream in real time to increase quality.

In the next pages, we will present in a detailed manner the algorithms above and their results.

2 Peer to peer file sharing protocols

2.1 The Gnutella protocol

Gnutella was one of the very first decentralized file sharing protocols to emerge. It was most popularly implemented in the file sharing application LimeWire. Gnutella is a great entrance point into gossip because it's one of the simplest gossip protocols that was used in production.

In Gnutella, the P2P swarm will itself handle search requests and peer discovery. The clients connect to each other through directly through a P2P overlay graph. The overlay graph is the P2P network that is "overlaid" on top of the actual underlying network. The underlying network in this case is IP itself—that's how nodes have to literally send packets to each other. Whenever the overlay network is insensitive to the underlying network topology, you're going to get suboptimal routing, since messages aren't taking the real world shortest path. More advanced P2P systems try to use smarter routing models which take the underlying network into account, but the simplest possible approach is to build an unstructured network, which produces its own random topology and follows that when it comes to message routing. Gnutella is an unstructured network.

In Gnutella, each node keeps track of a list of peers. Every node periodically pings its peers to ensure they're still online. If a node notices any of its peers have gone dark for too long, the node will un-peer with them and find someone else peer with. There are five message types in the Gnutella protocol:

Query	A search for a certain filename
QueryHit	A positive reply to a search, stating "Hey, I've got a file that matches that Query"
Ping	Probing a peer to see if they're alive
Pong	A reply to a Ping
Push	Requests a file be sent to the requester

The Ping and Pong are used for peer discovery and heartbeating. The Push message is only for coordinating file downloads when the file owner is behind a firewall.

The base of the protocol is Query and QueryHit. You want to disseminate your query widely, so you gossip this query to a random 3 of your peers. Each of those peers also gossip the query to a random 3 of their peers, and so on until a QueryHit response will appear.

How should we route the response back to the original sender? The solution that Gnutella uses is quite simple: route the response back recursively in order to maintain privacy and to not flood the network. Say you have a file that matches a Query for "metallica". You send back a QueryHit response back to whoever forwarded you that message, specifying the UUID you're responding to. By simply having each node remember who forwarded them each message, QueryHits can be recursively routed back to the originator. This minimizes privacy leakage and unnecessary noise in the network.

2.2 Gossip protocol

In a traditional gossip protocol, each node periodically gossips to K random targets. This K is known as the infection factor (as a nod to epidemiology). Once those targets receive the gossip, they randomly select another K targets to gossip to. This continues until either all reachable nodes receive the message, or the message expires.

If a node gossips to all of their peers, it's known as flooding. Bitcoin performs flooding rather than random infection.

Like many randomized protocols, gossip is imperfect, but it ends up approximating the properties of a minimum spanning tree (with high probability). At the same time, it offers much higher fault-tolerance.

Gossip protocols have several desirable properties for a P2P network:

Reliability (only a small fraction of intended recipients will not receive your broadcast)

Low latency $O(\log N)$ rounds for a message to disperse through the network

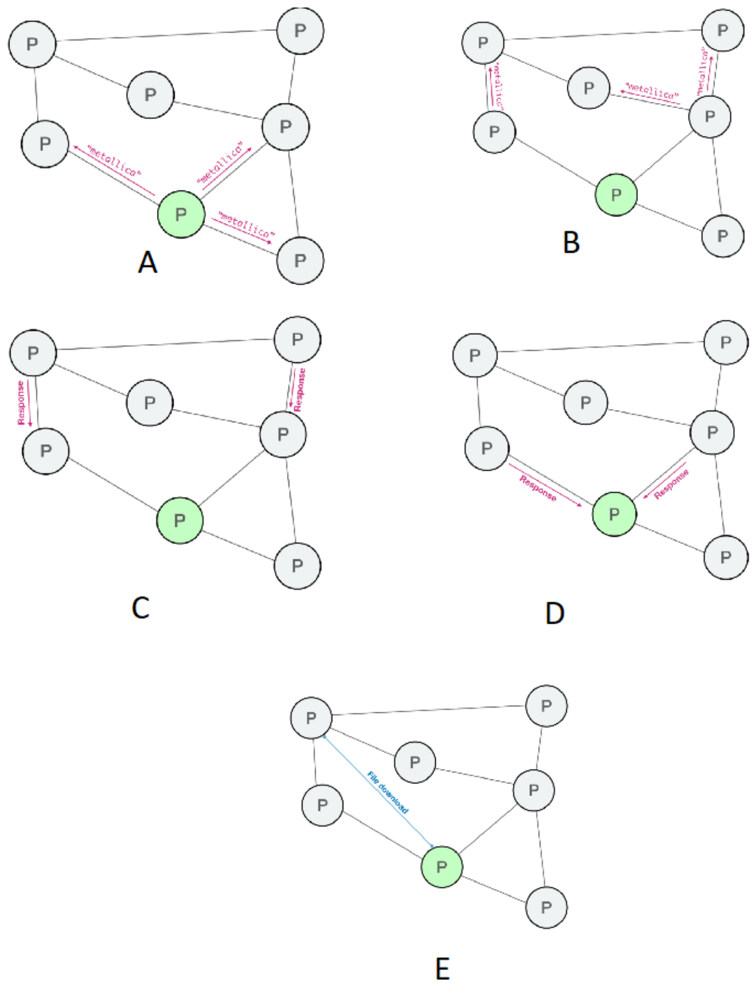


Figure 1: Gnutella architecture

Low message load per node

Very high fault-tolerance (at 50% node failure, it only requires twice as many rounds of gossip)

This makes gossip a prime candidate for message propagation in a system like blockchains.

2.3 Issues encountered in real world applications

A broadcast is sending a message to every node in a network. A multicast is when you want to send a message to many parties, but not everyone in the entire network. If you want to perform a search query in a P2P file sharing network, you actually don't want to send it to everyone in the network—if you did, the network would quickly get overloaded. Only a subset of the network needs to actually respond to our query.

The first thing you'd think of is probably just to iterate through everyone you want to reach and send them a point-to-point message.

This works, but it's not very efficient. It takes $O(N)$ time to perform the full multicast. It's also not realistic: it requires everyone in the network to maintain a list of every other node in the network. In a P2P system, this list can be huge and will change all the time due to node churn. (Plus, if the sender fails in the middle of this long multicast, the entire operation will fail.)

One way to optimize message propagation is by building a minimum spanning tree. A spanning tree is a tree that covers every node in a graph. A minimum spanning tree is the smallest possible such tree—in other

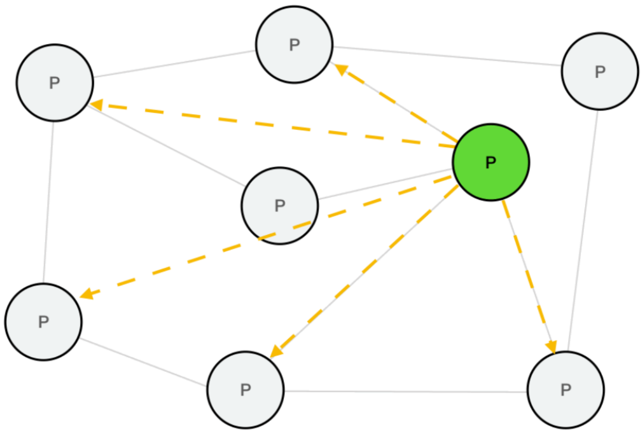


Figure 2: Simpel multicast

words, a spanning tree that is built with the minimum number of edges (or minimal total edge weights)

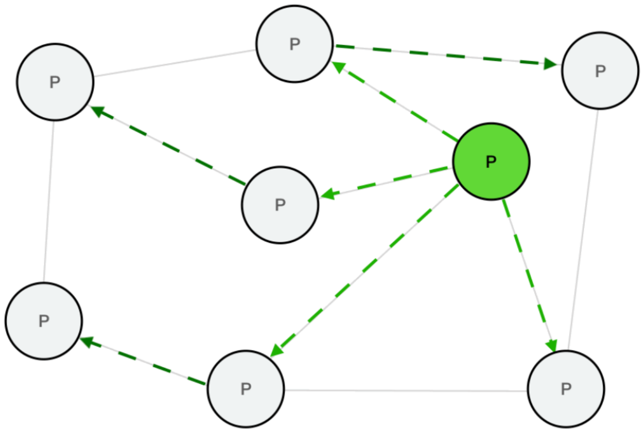


Figure 3: Minimum spanning tree

This should implement a broadcast in only $O(\log N)$ hops, since the messages will propagate in time proportional to the depth of the tree. This is theoretically optimal. A minimum spanning tree gives us maximally efficient routing, especially if the spanning tree is constructed in consideration of the underlying network topology. A spanning tree is great for when the network is static, but in a P2P network, it's a non-starter, we have to assume that nodes will crash, packets will be dropped, and the network topology will change over time.

If this forwarding process were to continue indefinitely, we'd have a problem: your message would loop around in the network forever. Node A would send to B, B would send to C,C would send back to A, and so on. To solve this, we'll give each message a UUID. By simply keeping track of UUIDs you've already forwarded, you can ignore repeat messages and thus prevent any infinite message loops.

Every message will propagate through the P2P network until it reaches literally everyone. This is fine if you want that, but it's overkill for file sharing. This would rapidly become a scalability bottleneck (every node would literally have to handle every single search in the whole network). We can add a TTL (time-to-live) on every message. The TTL is an integer that is decremented each time the message is forwarded.

Gnutella was not actually a great candidate for this kind of gossip-based network design. In Gnutella, more than half of the network received every single query. This is overkill for file sharing, where you don't need that large a portion of the network to receive each message—most searches are for common files, which many nearby nodes can serve you.

Bitcoin, on the other hand, requires every node to know about every block, and transactions are meant to propagate to everyone. Arguably, this makes Bitcoin better suited for a gossip-based protocol than Gnutella was.

2.4 How Torrents work and how they're different

BitTorrent networking protocol is the most popular protocol used in modern peer-to-peer file sharing. Unlike the traditional server to client file transfer methods, torrents work by gathering pieces of the file you want and downloading these pieces simultaneously from people who already have them downloaded. This means that the maximum speed obtained is based on the amount of people that have the file downloaded and not limited by a server as the traditional file transfer methods.

It's all about swarming and tracking, users download small pieces from different peers at once. As you add a torrent to your torrent client, your computer starts by connecting to the torrent's tracker servers, the tracker servers respond with a list of the users downloading the torrent or seeding (file fully downloaded but the client is still sharing it with other users) it. Afterwards, the client asks each client for a different piece of the torrent and it downloads it, as you download other peers ask your torrent client for some pieces that you've downloaded, and your client responds back by uploading those pieces to that peer. This process goes on till all the pieces are downloaded and put together. Afterwards your torrent client marks the torrent as finished and you're now seeding the file to other users.

The "*.torrent" file that you download from the internet contains information about the torrent trackers and how the torrent client should assemble the pieces that it's downloading together. The download speed is also controlled by the trackers, they monitor all the users downloading the torrent and reward the users who are sharing the file with others by increasing their allowed bandwidth, which means that if you try throttling down your upload speed, you'll only end up with less download speed.

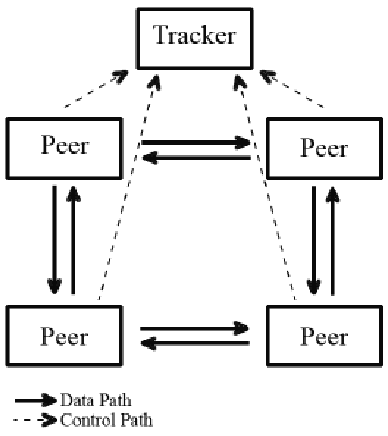


Figure 4: Torrent architecture

Swarming - splitting files into smaller pieces and sharing those pieces across a list of users.
Tracking - some servers help the torrent keep track of which users are downloading the torrent.

2.5 Methods used to encourege peers participation

Piece selection algorithm

Random first piece – There is no required selection and the process of uploading and downloading will start quickly.

Rarest first – In this the swarm selects the rarest piece of the file and starts downloading first, and replicates it in order to avoid the interruption of the slow downloading speed and also the loss of the file.

Strict Policy – If a piece of the file is requested then all the sub-pieces of a piece of the file need to download first and then the remaining files will download.

Endgame mode – When a piece of the file is requested for downloading then there is slow uploading of the file in order to ensure that the file needs to download first.

Once the download will finish the uploading speed boosts itself in order to provide the availability of the file for the swarm. These policies have to be implemented in order to avoid network failure or breaking of the swarm.

Choking

It's a tit-for-tat mechanism in order to get consistent download speed. It's a process of refusing the upload when the file is downloading. It helps in boosting the speed in order to download the files faster rate.

Optimistic Unchoking

It's not related to boosting the download speed, it's related to identifying the unused connection. The unused connection has to be eliminated or needs to be removed from the swarm so that, other peers will be available for the connection.

Anti-snubbing

It's again a good term in order to avoid the interruption of the network break. If a peer didn't receive anything from the last 60 seconds then it has to be considered as the snubbed peer.

Upload only

As the name suggests, if a peer completed the file download and there is nothing to download from the network then there is only one thing that happens called the upload.

2.6 Advantages and Disadvantages of torrents

Torrent are mainly used for distributing large files and popular files which have high traffic for relatively short periods, unlike traditional server/client downloads, high traffic leads to more efficient file sharing via BitTorrent. We cannot modify/update the file to newer versions once the torrent has been distributed, a new torrent with the updates needs to be created in order to share updates. The IP of all peers and info of files that are downloading are publicly available on trackers and the tracker is a critical component. If it fails, it can disrupt the distribution of all the files it has tracking and it is an easy distribution method for pirated/illegal content.

3 DONet

DONet is a Data-driven Overlay Network for live media streaming that relies on the data availability in order to guide the flow direction. The design of the streaming overlay is data-centric, a node always forwards data to others that are expecting the data, with no prescribed roles like father/child, internal/external, and upstreaming/downstreaming, etc.

The most important features of this design are the ease to implement as it does not have a complex global structures such as tree that needs to be constructed and maintained, the efficiency, as data forwarding is determined by its availability without any restrictions regarding directions and the robustness and resilience, as it allows adaptive and quick switching among multi-suppliers. The major advantage of this approach is that it does not rely on a semi-static structure that can become suboptimal due to node dynamics.

The core operations for a node in DONet are:

1. periodically exchange data availability information with a set of partners

2. retrieve unavailable data from one or more partners and supplies

3. supply available data to partners

3.1 Comparison with other peer to peer overlay systems

3.1.1 Tree-based overlay

Many overlay systems employ a tree structure, constructing and maintaining efficient distribution trees. For a large scale, the algorithms can also adopt hierarchical clustering to minimize the transmission delay. However, an internal node which always has a higher load can cause buffer underflow in its descendants. Even if there are tree repairing algorithms that can adapt to node dynamics, unbalanced load and vulnerability of the tree structure, these solutions often come with even more complex structures or advanced coding schemes which bring additional overhead.

3.1.2 Gossip-based overlay

We already talked about gossip protocols and how they have become popular solutions in multicast message dissemination in peer to peer systems. However, the use of gossip for streaming can cause significant redundancy because of its random push.

DONet uses a gossip protocol for membership management, but comes with a smart partner selection algorithm to intelligently pull data from multiple servers which helps with the redundancy.

3.2 DONet design

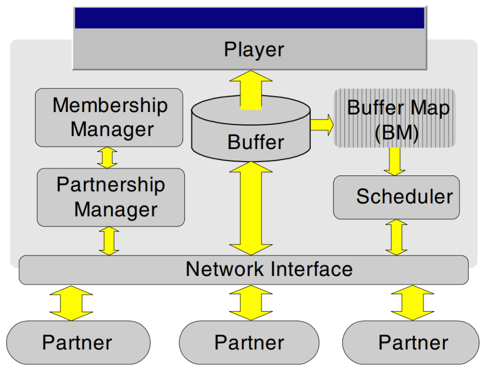


Fig. 1. A generic system diagram for a DONet node.

For each segment of a video, a node can be a receiver or a supplier or both depending on which information is available for this node. The source node is always a supplier and it is called the origin node. It can be a dedicated video server or a simple node that has a live video program to distribute.

3.2.1 Node Join and Membership Management

- Each node has a unique identifier (its ip address) and maintains a membership cache (mCache) with a partial list of identifiers for active nodes.
- A joining node contacts the origin node which randomly selects a deputy. The deputy will provide the new node with a list of contacts in order to establish partnerships
- In order to update the mCache, each node generates a membership message. SCAM (scalable gossip membership protocol) is used in order to distribute membership messages among the nodes.

3.2.2 Buffer Map Representation and Exchange

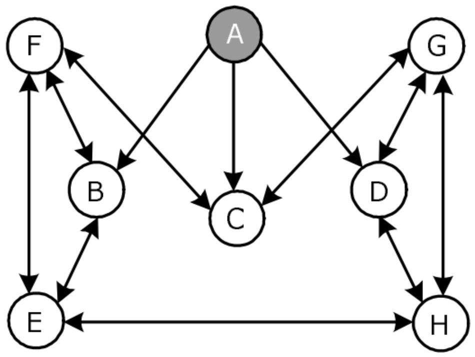


Fig. 2. Illustration of the partnership in DONet (origin node: A).

- A video stream is divided into multiple segments with the same length and the availability of the segments in the buffer is represented by a buffer map that is exchanged between partners.
- Each node schedules from what partner to fetch what segment based on the buffer map.

3.2.3 Scheduling Algorithm

The scheduling algorithm has two constraints: the playback deadline for each segment, and the heterogeneous streaming bandwidth from the partners. If the first constraint cannot be satisfied, then the number of segments missing deadlines should be kept minimum. This problem is NP-Hard (variation of Parallel machine scheduling) and as a solution, a heuristic of fast response time is used.

The heuristic algorithm calculates the number of potential suppliers for each segment based on the buffer maps of the partners. Since a segment with less potential suppliers is more difficult to meet the deadline constraints, the algorithm determines the supplier of each segment starting from those with only one potential supplier, then those with two, and so forth. Among the multiple potential suppliers, the one with the highest bandwidth and enough available time is selected.

A measure for preventing the origin node from being overwhelmed by requests (since it has all the segments available) is to advertise conservative buffer maps based on each partner.

3.2.4 Failure Recovery and Partnership Refinement

If a node departs gracefully, it should issue a departure message. If a node crashes, a partner that detects the failure, after an idle time or Buffer Map exchange, the partner will issue the departure message for the failed node. Each node will flush the entry from its mCache and it will reschedule if necessary.

Each node can periodically establish new partnerships with nodes randomly selected from its mCache. A node can also compute a partners score based on the maximum number of segments retrieved or received by the partner. After exploring new partners, the partner with the lowest score will be rejected in order to maintain a stable number of partners.

After experiments in the PlanetLab environments it was shown that the optimum number of partners is 4, having a balance between the control overhead and the continuity index. In terms of scalability, due to the fact that information is exchanged locally between partners, the control overhead of each node is independent of the total number of nodes and the continuity even increases due to increasing cooperation.

3.3 Comparison with Tree-based Overlay

Now, compared to a Tree-based overlay with the same degree as a DONet node (3 children and a parent node), using the average hop count to compare end to end delays, the tree based actually performed worse (because the tree becomes unbalanced).

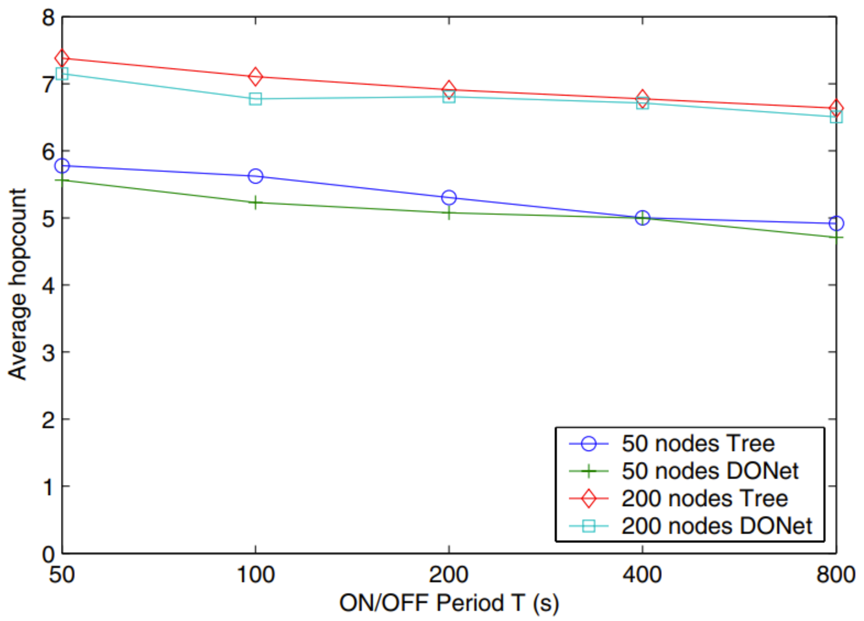


Figure 5: Average overly hop-count of DONet and tree-based overlay

As in terms of continuity, the tree not only performs worse, but also has fluctuations due to the buffer underflow mentioned in section 2.1.1.

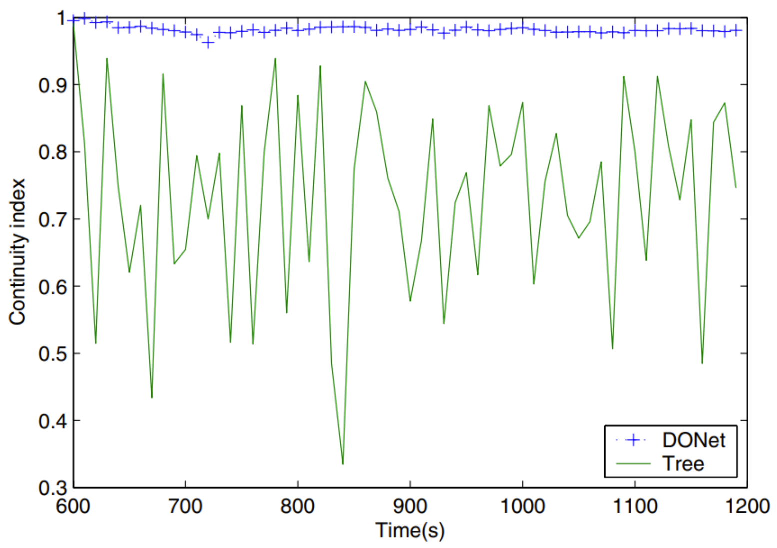


Figure 6: Samples of continuity indices for DONet and a tree-based overlay

3.3.1 Cool Streaming

DONet also has an implementation called CoolStreaming v.0.9, released on May 30, 2004, which has attracted over 30000 distinct users with more than 4000 simultaneously being online at some peak times. Some discoveries that came with this implementation were the fact that the internet (at that time) has enough bandwidth to support TV-quality streaming and the fact that a larger data-driven overlay leads to a better streaming quality and continuity.

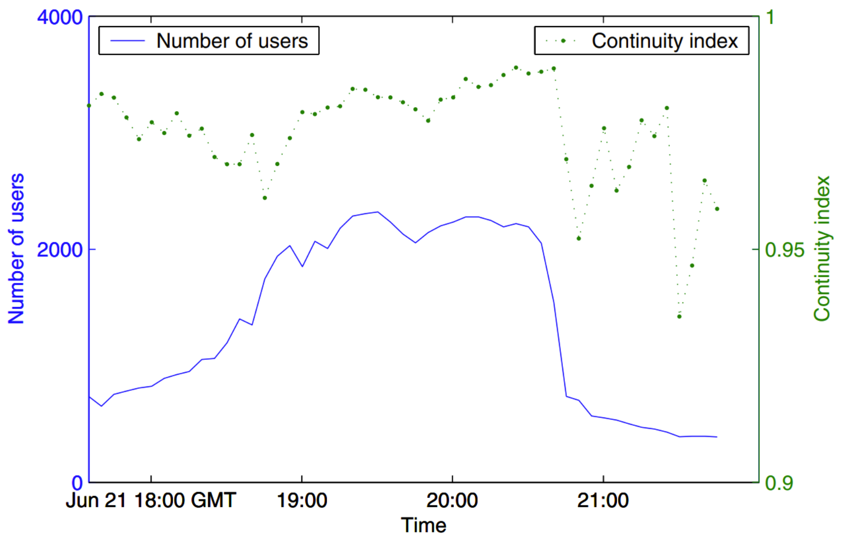


Figure 7: Number of users and Continuity index over time

In the next pages, we will present a protocol that is using encoding of geopositioning in real time for audio/video streaming for mobile devices.

4 P2P Audio/Video Protocol with Global Positioning Data in Real Time for Mobile Devices

(1) Applications that are using Peer to Peer communication to interact via audio/video can suffer problems because of computing power and available bandwidth. This issue is more visible as the number of users is increasing. All the current content sharing services are using geopositioning for being able to transmit securely the location of the audio/video stream in real time without losing quality. To encode the global positioning coordinates, these applications are using longitude and latitude or Universal Transverse Mercator (UTM). This technique can face problems when trying to concatenate the coordinates for multiple nodes which leads to waste of precision. For that the transmission bandwidth and the data size are relevant. In "P2P Audio/Video Protocol with Global Positioning Data in Real Time for Mobile Devices"(1) the authors are proposing a technique for adding a geopositioning signal corresponding to the N participants in a multi-party videoconferencing, so that the accuracy is tailored to the bandwidth unused by the audio/video channels, without producing delay. They tried to develop a lossy positioning information compression technique with minimum precision required using the range of probable positions and the number of bits available for geopositioning information. Their P2P audio/video stream processing system is balancing the network and computational resources that are loaded in all the machines involved in MVoIP communication. In their approach the system is fully distributed and self-organizing, also each client has to perform a third of the transmission and half of the operations which gives the system a fair load distribution of data mixing and transmission actions. This protocol is implicitly producing the audio distribution so that when the mixing phase ends there is no audio distribution to make because all the machines already have the audio. The authors are claiming that "It is an adequate protocol for communicating two or more machines of limited resources (mobile phones or PDAs for example) without employing a specialized server or promoting one of the machines as server"(1).
Notations in the paper:

- A devices ring - a subset of devices with modular sequential order and some characteristics in common
- R_R - real device ring
- R_C - connected devices ring
- N_R - total number of machines in R_R
- N_C/N - total number of machines in R_C
- n - current machine in R_C
- I - total number of iterations of the algorithm
- i - current iteration of the algorithm
- $Mix_{y=0}^{x}(V_y)$ - a packet mixing function that mixes from $y = 0$ to x
- V_n - the current voice packet of node n
- V_a - e fully mixed voice packet ready for playback
- Parallel {{Job1}} {{Job2}} - job1 and job2 are carried out in parallel

4.1 Position Coordinates Encoding

This system will need to place a number of nodes in their geographical locations, but if the distance between them is greater, then it is less relevant. The meaning of this is the fact that a general view of the location of each node that will be seen as a set in a mobile device is given by the location signal to be transmitted. To achieve this, the authors has defined an action window to represent all nodes location with a level of precision suitable to the visualization capabilities and free bits for transmission available.

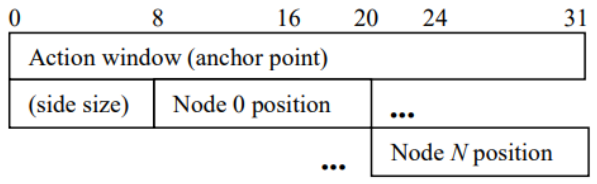
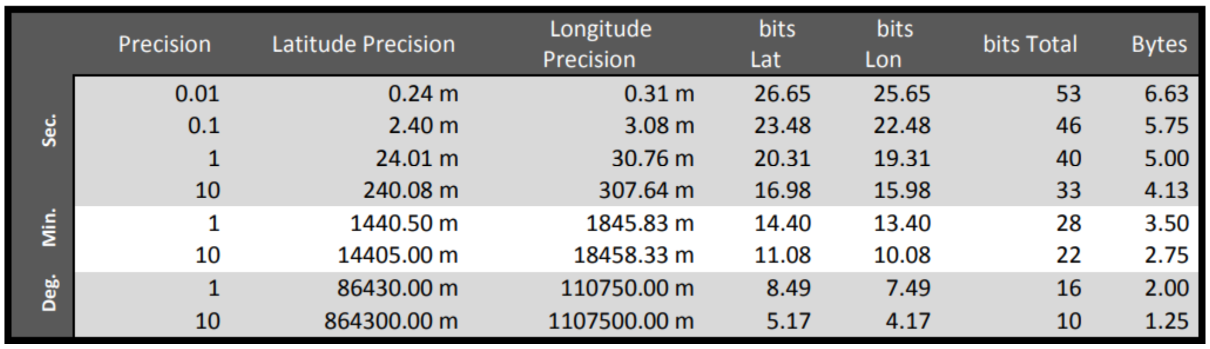


Figure 8: Encoding bit arrangement

The action windows is representing an area of the Earth surface that contains the location of all nodes, but, in order to define it, is not required a good precision, but the ability to adapt to large or small scales. It is defined by it's anchor point and length. The area of the surface is specified by the latter, in this way the number of bytes required to represent it is reduced.(Check: 9)
The last four bits in the encoding of the anchor point are used to encode the scale in which the square side



	Precision	Latitude Precision	Longitude Precision	bits Lat	bits Lon	bits Total	Bytes
Sec.	0.01	0.24 m	0.31 m	26.65	25.65	53	6.63
	0.1	2.40 m	3.08 m	23.48	22.48	46	5.75
	1	24.01 m	30.76 m	20.31	19.31	40	5.00
	10	240.08 m	307.64 m	16.98	15.98	33	4.13
Min.	1	1440.50 m	1845.83 m	14.40	13.40	28	3.50
	10	14405.00 m	18458.33 m	11.08	10.08	22	2.75
Deg.	1	86430.00 m	110750.00 m	8.49	7.49	16	2.00
	10	864300.00 m	1107500.00 m	5.17	4.17	10	1.25

Figure 9: Byte and precision trade off for latitude and longitude in an area 40ºN from the equator.

length will be specified in the next byte. (Check: 10)
The precision level must always be less than the maximum achievable precision. With this in mind, the authors defined the maximum precision as "the one with which a coordinate can be selected within the action window, visualized in full screen on a mobile device with a maximum resolution of 640x480 pixels"(1). In this case the maximum achievable precision is the size of the action window divided by 640.
We consider now a grid of 64×64 elements. Each node will be placed in the action window so that we can determine its position within an n by m elements grid.

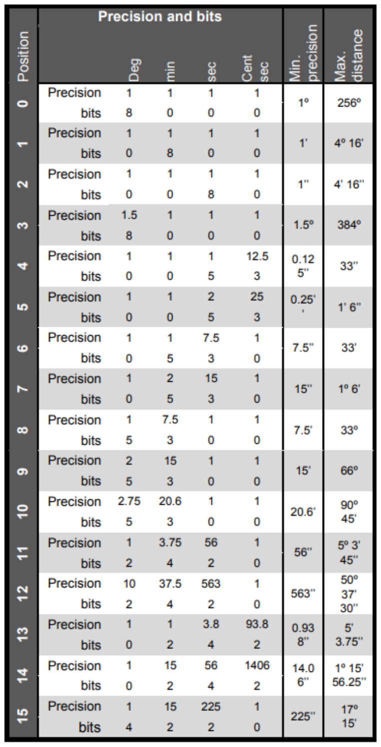
4.2 Protocol Definition

"This protocol is defined as a packet mixing and distribution algorithm in a network of N machines."(1) The general algorithm covers the case when N machines are connected denoted from 0 to $N-1$ so the emitting and receiving nodes will be:

$$N_e(n,i) = n + 2^{i-1} mod N \tag{1}$$

and

$$N_r(n,i) = n - 2^{i-1} mod N \tag{2}$$



Position	Precision and bits	Deg	min	sec	Cent sec	Min. precision	Max. distance
0	Precision	1	1	1	1	1º	256º
	bits	8	0	0	0		
1	Precision	1	1	1	1	1'	4º 16'
	bits	0	8	0	0		
2	Precision	1	1	1	1	1''	4' 16''
	bits	0	0	8	0		
3	Precision	1.5	1	1	1	1.5º	384º
	bits	8	0	0	0		
4	Precision	1	1	1	12.5	0.12 5''	33''
	bits	0	0	5	3		
5	Precision	1	1	2	25	0.25' '	1' 6''
	bits	0	0	5	3		
6	Precision	1	1	7.5	1	7.5''	33'
	bits	0	5	3	0		
7	Precision	1	2	15	1	15''	1º 6'
	bits	0	5	3	0		
8	Precision	1	7.5	1	1	7.5'	33º
	bits	5	3	0	0		
9	Precision	2	15	1	1	15'	66º
	bits	5	3	0	0		
10	Precision	2.75	20.6	1	1	20.6'	90º 45'
	bits	5	3	0	0		
11	Precision	1	3.75	56	1	56''	5º 3' 45''
	bits	2	4	2	0		
12	Precision	10	37.5	563	1	563''	50º 37' 30''
	bits	2	4	2	0		
13	Precision	1	1	3.8	93.8	0.93 8''	5' 3.75''
	bits	0	2	4	2		
14	Precision	1	15	56	1406	14.0 6''	1º 15' 56.25''
	bits	0	2	4	2		
15	Precision	1	15	225	1	225''	17º 15'
	bits	4	2	2	0		

Figure 10: Different encoding for the action window side size

The mixing and distribution characteristics of the algorithm are defined by:

$$P_e(n,i) = Mix_{y=0}^{2^{i-1}-1}(V_{(n-y)modN}) \tag{3}$$

where $P_e(n,i)$ is the the composition of the packet that node n will have to send during iteration i.

$$P_r(n,i) = Mix_{y=0}^{2^{i-1}-1}(V_{((n-2^{i-1})-y)modN}) \tag{4}$$

where $P_r(n,i)$ is the the composition of the packet that node n will receive in iteration i.

$$P(n,i) = Mix_{y=0}^{2^{i}-1}(V_{(n-y)modN}) \tag{5}$$

where $P(n,i)$ is the final packet that node n will have composed after the reception of the last packet during iteration i

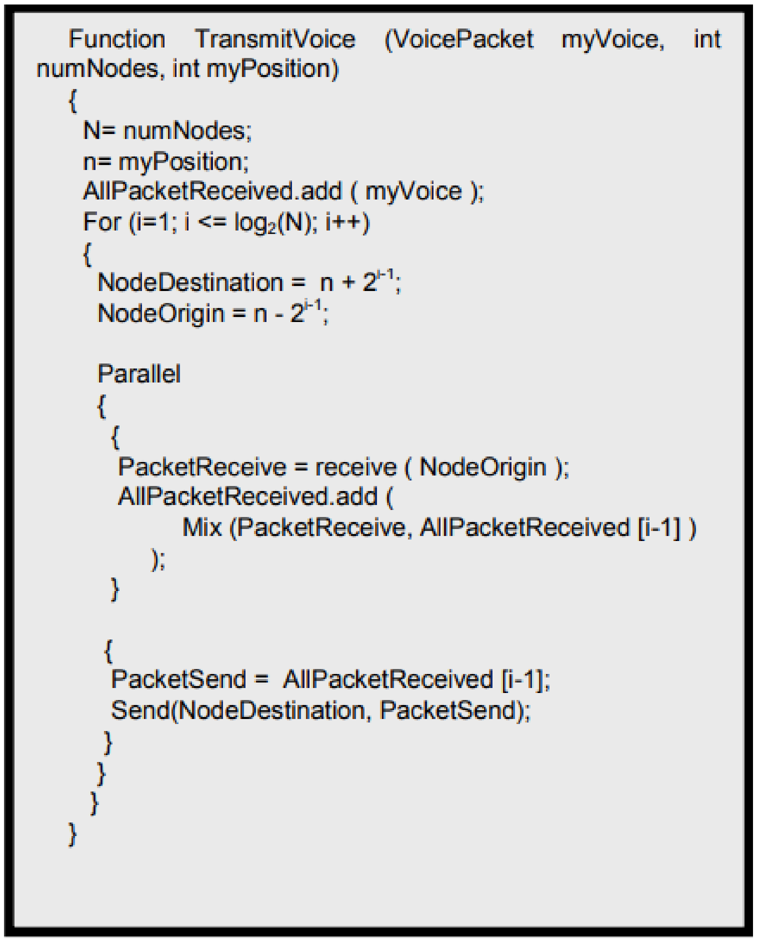
$$V_a(n,i) = Mix_{y=1}^{2^{i}-1}(V_{(n-y)modN}) \tag{6}$$

where V_a is the accumulated composition of the voice packet for playback at node n during iteration i

$$DV_a(n,i) = Mix_{y=1}^{N-1}(V_{(n-y)modN}) \tag{7}$$

where DV_a is the desired accumulated composition of the voice packet for playback at node n during iteration i.

The equations 3 to 7 can be defined in a recursive way using the relationships between them and $N_e(n,i)$ and $N_r(n,i)$ from 1, 2.
Using 6, it can be developed an algorithm to represent the audio packets mixed for playback in node n when there are a total of N machines.(Figure 11). It is representing the values of V_a for the node $n = N-1$ and the values V_k decreasing from $N-2$ to 0.
The results of this algorithm are showing that there are three correspondence functions between V_a and DV_a:



```
Function  TransmitVoice  (VoicePacket  myVoice,  int
numNodes, int myPosition)
  {
   N= numNodes;
   n= myPosition;
   AllPacketReceived.add ( myVoice );
   For (i=1; i <= log2(N); i++)
   {
    NodeDestination =  n + 2^(i-1);
    NodeOrigin = n - 2^(i-1);

    Parallel
    {
     {
      PacketReceive = receive ( NodeOrigin );
      AllPacketReceived.add (
              Mix (PacketReceive, AllPacketReceived [i-1] )
          );
     }

     {
      PacketSend =  AllPacketReceived [i-1];
      Send(NodeDestination, PacketSend);
     }
    }
   }
  }
```

Figure 11: General Algorithm

1. $N = 2^I$
   This case follows directly from the general algorithm and if the number of users is between 1 and k, the probability to reach this case is $[log_2(k)]/k$

2. $N <> 2^I$ and $N = 2^{I-1} + 2^x$ where $x < 1$
   In this case, the formula for P_e must be changed from $P_e(n,i) = P(n,i-1)$ in:

$$P_e(n,i) = \begin{cases} P(n,i-1) & \text{if} i <> I \\ P(n,x) & \text{if} i = I \end{cases} \tag{8}$$

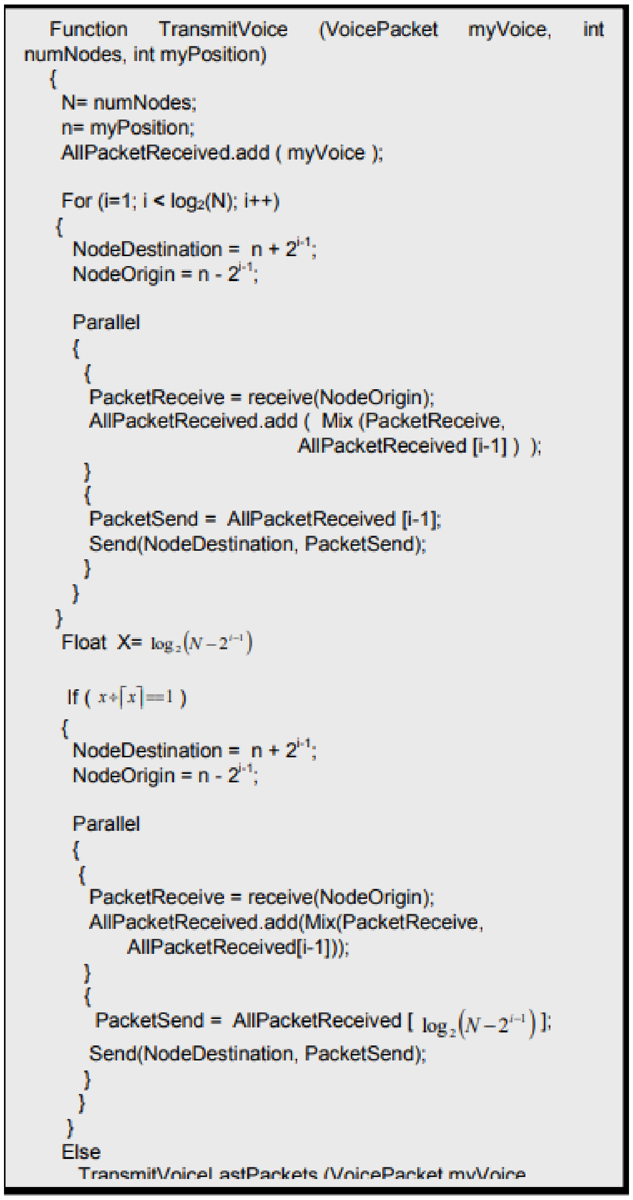
   where

$$x = log_2(N - 2^{I-1} \tag{9}$$

   The probability for this case to occur is $\frac{\sum_{x=1}^{[log_2(k)]}(x-1)}{k}$.

3. $N <> 2^I$ and $N <> 2^{I-1} + 2^x$ where $x < I$
   In this case, in order for V_a to be the same as DV_a is necessary to add iterations to transmit the required packet size. In the worst case, the number of iterations is between $([log_2(k)] - 1) - 1$ and $[log_2(k)]$. The probability for this would be $\frac{[log_2(k)]}{k}$. In the normal cases the number of iterations is in the interval $[1, ([log_2(k)] - 1) - 2]$. The probability would be $\frac{k - \sum_{x=1}^{[log_2(k)]}(x+1)}{k}$

After the algorithm is changed in order to cover all the cases, it will look like Figure 12.



```
  Function    TransmitVoice    (VoicePacket    myVoice,    int
numNodes, int myPosition)
  {
   N= numNodes;
   n= myPosition;
   AllPacketReceived.add ( myVoice );

   For (i=1; i < $\log_2$(N); i++)
   {
     NodeDestination =  n + $2^{i-1}$;
     NodeOrigin = n - $2^{i-1}$;

     Parallel
     {
      {
       PacketReceive = receive(NodeOrigin);
       AllPacketReceived.add (  Mix (PacketReceive,
                                  AllPacketReceived [i-1] )  );
      }
      {
       PacketSend =  AllPacketReceived [i-1];
       Send(NodeDestination, PacketSend);
      }
     }
   }
   Float  X= $\log_2(N-2^{i-1})$

    If ( $x \div \lceil x \rceil == 1$ )
   {
     NodeDestination =  n + $2^{i-1}$;
     NodeOrigin = n - $2^{i-1}$;

     Parallel
     {
      {
       PacketReceive = receive(NodeOrigin);
       AllPacketReceived.add(Mix(PacketReceive,
            AllPacketReceived[i-1]));
      }
      {
        PacketSend =  AllPacketReceived [ $\log_2(N-2^{i-1})$ ];
       Send(NodeDestination, PacketSend);
      }
     }
    }
   Else
     TransmitVoiceLastPackets (VoicePacket myVoice
```

Figure 12: Final Algorithm

In order to incorporate geopositioning to the data streaming, the protocol was modified by adding reception of each nodes position during the stream establishment phase (in this way, the node who started the communication will compute the action window parameters). Moreover, each node encapsulates the geopositioning data that is available on each iteration within the audio/video data in the sub-mixing phase. Also, if the action window parameters are requiring a dynamic change, the starting node will be the one who will send the corresponding control signal encapsulated within the stream.

4.3 Results

Because this protocol is using encapsulation for the geopositioning data and it is transmitting it using less than 50 bytes, the quality of audio/video is not affected.
This encoding scheme is the only one that is never using more information than is representable (Figure 13) because the other schemes are using a fix precision.

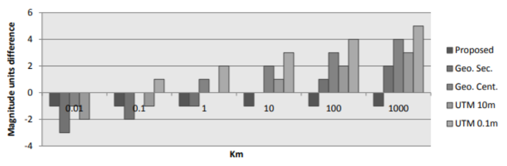


Figure 13: Magnitude units difference betweendata precision and maximum achievable precision

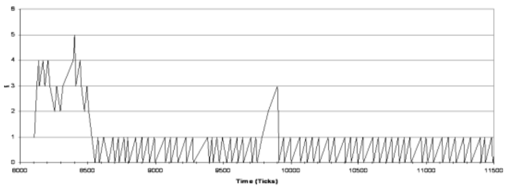


Figure 14: Buffer State

In figure 14 and 15 it can be seen the way that this protocol evolves when two nodes are making a video/voice stream. In the first one is represented the buffer state while the stream is on and in the second figure are represented the lost packages during the stream. It can be observed that the number of lost packages is low, reaching 0 in the interval $[11, 15]$.

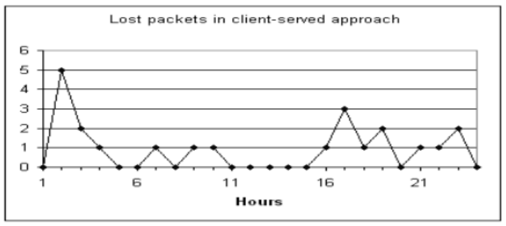


Figure 15: Number of lost packets ina 24h hours conversation

5 Conclusion

As we have presented peers needs an incentive to contributed their physical resources (eg. CPU, memory, band-with etc.). Early P2P protocols failed because 80% of users didn't contribute anything to the network. Torrent manege to offer easy and efficient access to highly demanded files and it's model punished users that didn't contribute to the network by limiting access to the networks resources. Illegal distributing sites like Filist also apply a quota for users by implementing a "pay in GB upload". We can see that some variation of Gnutella gossip protocols are used in blockchain networks where miners have a monetary incentive to keep the network healthy. We have found in many articles that the old P2P protocols that failed because of lack of resources can be adapted and used with great efficiency in the modern distributed world, in private Cloud or Blockains as well as public application if users are stimulated to share resources.

We also saw that a data-driven overlay which uses a gossip protocol for membership distributions and an intelligent scheduling algorithm can support TV-quality (450Kbs) live streaming. The partnership approach proved to be scalable with bounded delay while also maintaining a high continuity index that even improves with the number of nodes, these performances exceeding those of a tree-based overlay.

José-Vicente Aguirre, Rafael Álvarez, Leandro Tortosa, and José-Francisco Vicent with "P2P Audio/Video Protocol with Global Positioning Data in Real Time for Mobile Devices" managed to develop an original way to add audio/video signal geopositioning capabilities in real time by encoding the global positioning coordinates without affecting the performance. It is easily to extend to support new requirements and it is suitable when is needed a list of positions within a designated area to encode with an adjusted degree of precision.

This protocol is the only one that is using an encoding scheme that is never using more information than needed, being interpreted in a dynamic manner. This fact is not affecting the quatily of the stream, the number of lost packages being low.

References

[1] José-Vicente Aguirre, Rafael Álvarez, Leandro Tortosa, and José-Francisco Vicent, P2P Audio/Video Protocol with Global Positioning Data in Real Time for Mobile Devices.

[2] CoolStreaming/DONet: A Data-driven Overlay Network for Peer-to-Peer Live Media Streaming

[3] Recent Advances in Peer-to-Peer Media Streaming Systems

[4] Gnutella: an Intro to Gossip

[5] State-of-the-Art Techniques for P2P Traffic Control on Corporate Networks:A Review

[6] Algorithm Used In BitTorrent, uTorrent

[7] How do torrents work? (P2P networking)