A Fault-Tolerant P2P-based Protocol for Logical Networks Interconnection

Jaime Lloret[1], Juan R. Diaz[2], Fernando Boronat[3] and Jose M. Jiménez[4]
Department of Communications, Polytechnic University of Valencia (Spain)
[1]jlloret@dcom.upv.es; [2]jrdiaz@dcom.upv.es; [3]fboronat@dcom.upv.es; [4]jojiher@dcom.upv.es

Abstract

TCP/IP protocol suite application layer allow grouping nodes into logical networks according to similar characteristics, functionalities or social trends. Sometimes, it is useful, or necessary, to find a way to join those autonomous logical networks. This paper describes how a group of nodes, with the same role in a network, will establish logical connections with other nodes from other groups. This architecture allows sharing data, content and resources between different groups. The topology of logical connections, between nodes from different groups, changes depending on their processing, the number of connected nodes or node failures. It is scalable and fault-tolerant. The protocol and its management are described in detail. Simulation results for some cases are shown. The architecture has been developed to be applied to multiple types of logical networks (P2P file-sharing, distributed computing, CDNs and so on).

1. Introduction

Logical network group nodes using the same application layer protocol. All nodes use that protocol to share their resources (e.g. content, files, computing, etc.) with others and to reach other nodes in the network. Let's suppose a scenario where there are several logical networks. All nodes in the same network have the same role in the network. Application layer protocols running in each network can be translated to other protocols. So, the protocol shouldn't be encrypted using signatures or private keys. Examples of networks with these are some P2P file-sharing networks, content delivery networks, distributed computing networks, and so on.

Researchers in this topic focus their work in a centralized manner, e.g. using a central point of data delivering between logical networks. But this solution suffers from three main drawbacks: (1) All protocols have to be running in the server at the same time, so it will need many resources. (2) There is a single point of failure and a bottleneck. (3) Every time a new network is joined, the central point has to be upgraded.

We propose the use of a common protocol to join nodes from different networks. Every node knows the protocol of its network and the protocol for the interconnection (only two protocols). Nodes in a network could have connections with nodes from other networks based on some parameters such as available capacity, available number of connections, and so on. This architecture allows several layers to organize node connections. When a new network joins the interconnection system, those new nodes will use the proposed protocol to have connections with nodes from other networks instead of upgrading all networks' nodes. The proposed architecture allows node's self-organization and it is scalable. Moreover, fault-tolerance and load balancing could be implemented.

This paper is structured as follows. Section 2 describes the architecture. Section 3 explains the protocol operation. Algorithms to recover node failures and leavings are described in section 4. Simulations for some cases are shown in section 5. Finally, in section 6, the conclusions are summarized.

2. Architecture description

The architecture uses some nodes from existing logical networks to interconnect them. When a node looks for some information, first it will try to get it from its network. In case of no result, the search is sent to other networks using the proposed protocol. If it is found, the information can be downloaded. Once the node has the information, it will act as a cache for its network, sharing this information.

We have defined 3 types of node roles. A node could run all them simultaneously.

- Dnode: They have connections with Dnodes from other networks. Dnodes are used to send searches and data transfers between networks as a hub-and-spoke.
- Level-1 Onodes: They organize Dnodes in zones to have a scalable architecture.

- Level-2 Onodes: They have connections with Level-2 Onodes from other networks. Level-2 Onodes organize connections between Dnodes from different networks.

Onodes maintain and manage the architecture. Every new Onode must authenticate with other Onodes from its network and/or from others networks (depending on the Onode). When a new node joins the proposed architecture, it starts with its upstream and downstream bandwidth, its maximum number of supported connections from other nodes and its maximum % of CPU load used for joining the architecture by the desktop application. All nodes have a unique node identifier (nodeID), first node in the network will be Dnode, level-1 Onode and level-2 Onode and it will have nodeID=1. Then, it will assign nodeIDs sequentially to new nodes in its network. All networks have a network identifier (networkID) and all nodes in the same network have the same networkID.

We define λ parameter as the node's capacity. It depends on node's bandwidth (in Kbps), its number of available connections, its maximum number of connections and its % of available load. It is used to determine the best node to have connections with.

We define δ parameter as the node suitable parameter. It depends on node's bandwidth and its age in the system. It is used to know which node is the best node to have higher role. Nodes with higher bandwidth and older are preferred to promote, so they will have higher δ. Every β nodes in the logical network, the node with higher δ will start Dnode role and it will authenticate with a level-1 Onode in its network. In order to have a scalable network, every α Dnodes, the node with higher δ will start level-1 Onode role and it will create a new Dnodes' zone. α and β values depend on the number of nodes in the network and their data traffic. Using explanations given in [1], we have estimated α between 64 and 96, β between 1 and 96 for P2P File Sharing Networks, and one level-2 Onode every 50 level-1 Onodes is enough to maintain and manage level-1 Onodes' network.

We have chosen SPF (Shortest Path First) routing algorithm [2] to route information between Onodes. It is fast and allows sending fast searches to find Dnodes adjacencies, but it can be changed for other routing protocol depending on the networks' characteristics. Both type of Onodes use SPF algorithm. Level-l Onodes routes information inside the network using NodeID values. Level-2 Onodes routes information between networks. NetworkID is used to route information in this layer. Every node runs SPF algorithm locally and selects the best path to a destination based on a metric described later. Level-1 Onodes only add level-2 Onodes in their network entries and level-2 Onodes add all level-1 Onodes in its network, so they know how to reach all level-1 Onodes in their network. SPF routes are calculated using the virtual-Link cost which is based on node's capacity (λ). The more is the node's capacity, lower its cost is. The metric for each route is based on the hops to a destination and the virtual-link cost of those nodes involved in the path. The metric is used in SFP algorithm to know which is the best path to reach a node is. More information about the description of this layered architecture can be found in [1].

3. Protocol description

When the first network node joins the architecture, it has to send a discovery message, with its networkID, to find level-2 Onodes from other networks. If there is not any reply in a certain period of time, it will begin the process again. Level-2 Onodes from other networks reply this message with their networkID and their λ parameter. It chooses level-2 Onodes with higher λ and sends them a connection message. Then, they reply with a welcome message indicating that it has joined the architecture. After that, it sends them its neighbor list. Its neighbors add this entry to their topological database and recalculate routes using SPF algorithm. When they finish, they will send their database to the new level-2 Onode to build its database. Next database messages will be updates only. Finally, it will send them keepalive messages periodically to indicate that it is still alive. If it does not receive a keepalive message from a neighbor for a dead time, it will erase this entry from its database. Steps explained are shown in figure 6.

When new nodes join the architecture, they will be Dnodes. At first, a Dnode sends a discovery message with its networkID to level-1 Onodes known in advance or by bootstrapping [3]. Only level-1 Onodes with the same networkID will reply with their λ parameter. Dnode will wait for a hold time and choose the level-1 Onode with higher λ. If there is no reply for a hold time, it will send the discovery message again. Next, Dnode sends a connection message to the elected level-1 Onode. This level-1 Onode will reply a welcome message with the nodeID assigned and information related with the backup level-1 Onode. Then, it will add Dnode's entry to its Dnodes' table. Finally, Dnode will send keepalive messages periodically to the level-1 Onode. If the level-1 Onode does not receive a keepalive message from the Dnode for a dead time, it will erase this entry from its database. Steps explained can be seen in figure 7.

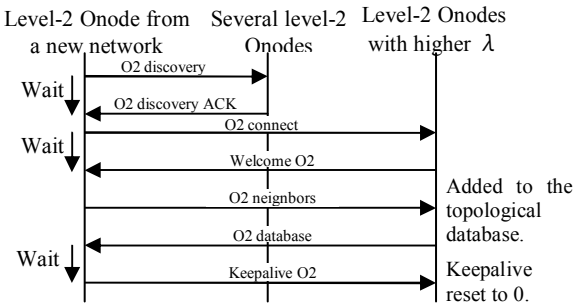

Figure 6. Messages when enters a new network.

Dnodes have to send request messages to the level-1 Onode in its zone to establish connections with Dnodes from other networks. This message contains a 0x00 value (or a destination networkID), sender's networkID, sender's nodeID and its network layer address. Level-1 Onode routes it to its nearest level-2 Onode. Level-2 Onode sends this request to other networks (or network).

When a level-2 Onode receives this message from other network, it will send a message to level-1 Onodes in its network in order to find Dnodes with highest λ in the network. Every request has a unique sequence number to avoid route loops in the network. Level-1 Onodes will reply with their 2 Dnodes with highest λ. Level-2 Onode waits replies for a certain period of time. It chooses 2 highest λ Dnodes and sends them a message. The highest one will be the preferred; the second one will act as a backup. This message contains the nodeID and the requesting Dnode's network layer address. When these Dnodes receive that message, they will send a message to connect with the Dnode from the other network. Next, they send a confirmation message to the level-2 Onode in its network to indicate a connection has established with other network Dnode. If level-2 Onode does not receive this message for a hold time, it will send a new message to the next Dnode with highest λ. This process will be repeated until level-2 Onode receives both confirmations. When the requesting Dnode from other network receives these connection messages, it will add Dnode with highest λ as its first neighbor and the second one as the backup. Then, it replies these connection messages to acknowledge the connection. If the requesting Dnode does not receive any connection from other Dnode for a dead time, it will send a requesting message again. Finally, both Dnodes will send keepalive messages periodically. If a Dnode does not receive a keepalive message from the other Dnode for a dead time, it will erase this entry from its database. Every time a Dnode receives a search or data transfer for other networks, it looks up its Dnodes' distribution table to know which Dnode send the search, or data, to. These steps are shown in figure 8.

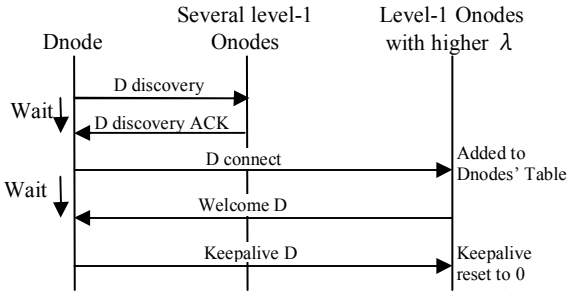

Figure 7. Messages when a new Dnode joins the architecture.

When a level-2 Onode receives a new networkID in its networkID table, it will send a message to all level-1 Onodes in its network with a sequence number to avoid route loops. Then, level-1 Onodes will forward this message to all Dnodes in their zone. Subsequently, Dnodes will begin the process to request Dnodes from the new network as it is shown in figure 8.

When a level-2 Onode calculates that the average of λ for all level-1 Onodes in its network accomplishes equation 1, it will send a message to all level-1 Onodes to request a new level-1 Onode.

$$\lambda \leq 1.05 \frac{k_3}{Max\_Con} \tag{1}$$

We have considered $K_3=10^3$ to get λ into desired values. When a level-1 Onode receives that message, it will reply with the nodeID of the Dnode with highest δ in its zone. Level-2 Onode will process all replies and will elect the Dnode with highest δ. Then, it will send a message to the level-1 Onode with highest δ in its network. This message will be routed to the chosen Dnode. This Dnode will become a level-1 Onode and will send a disconnection message to its level-1 Onode. If the level-2 Onode does not receive changes for a hold time, it will send a new request message to the second Dnode with highest δ. If this time it fails again, it will begin again, but avoiding those Dnodes. Steps explained are shown in figure 9.

Level-1 Onodes must authenticate with level-1 and/or level-2 Onodes in their network. A new Onode can establish its first connection with any Onode known in advance or by bootstrapping [3]. First, it sends a discovery message with its networkID. Only level-1 Onodes with the same networkID will reply with their λ. New level-1 Onode will wait for a hold time and will choose the Onodes with highest λ. If there is no reply, new level-1 Onode will send a discovery message again. Then, new level-1 Onode will send a connection message to the chosen level-1 Onodes. They will reply with a welcome message indicating it is connected to the architecture and they will become its neighbors.

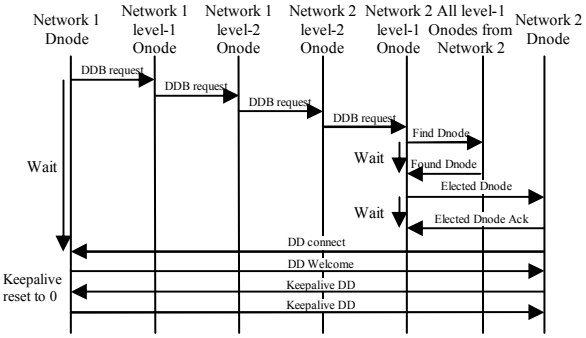

Figure 8. Dnode message to request connections.

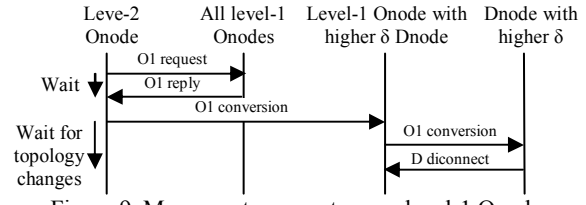

Figure 9. Messages to request a new level-1 Onode.

New level-1 Onode will send its neighbor list to all of them to update their level-1 Onode network database and all of them will recalculate new routes using SPF algorithm and the metric aforementioned. Then, they send their database to the new level-1 Onode in order to build its level-1 Onode network database. Next times it will only receive updates. New level-1 Onode will send keepalive messages to its neighbors periodically. If the level-1 Onode does not receive a keepalive message from its neighbor for a dead time, it will erase this entry from its database. Steps explained are shown in figure 10.

First level-2 Onode, that checks equation 2, will send a message to generate a new level-2 Onode.

$$\frac{number\_of\_level-1\_Onodes}{Max\_Con \bullet number\_of\_level-2\_Onodes} \geq 1 \qquad (2)$$

This message will be sent to the level-1 Onode with highest δ. When the level-1 Onode receives that message, it will begin to use level-2 organization layer functionalities. Then it sends a message to its neighbours to inform them about it. If the level-2 Onode initiating this process does not receive changes in its level-1 Onode's table for a hold time, it will send a new request message to the second level-1 Onode with highest δ. Then, new level-2 Onode will proceed with the same manner as it has been explained in figure 6. Figure 11 shows steps explained.

4. Leavings and Fault Tolerance

Every level-2 Onode sends its backup information to the highest δ level-1 Onode in the network

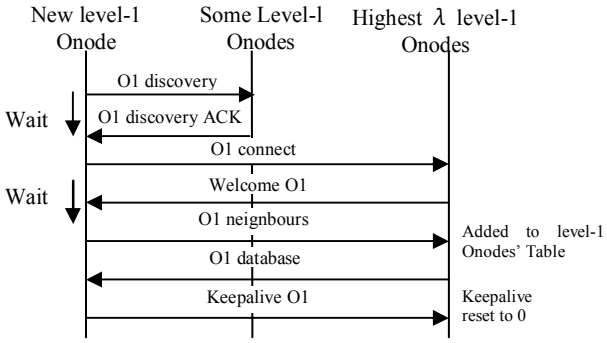

Figure 10. Messages when enters new level-1 Onode.

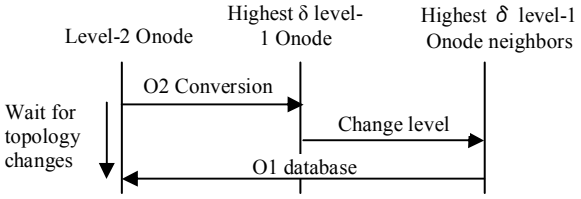

Figure 11. Messages to request a new level-2 Onode

periodically. When a level-2 Onode leaves the architecture voluntarily, it will send a message to the highest δ level-1 Onode announcing it. The level-1 Onode becomes a level-2 Onode and acknowledges that message. Then, level-2 Onode leaves the architecture sending a disconnection message to its neighbors. If that level-2 Onode does not receive the acknowledgement, it will begin the process with the second highest δ level-1 Onode. Next, new level-2 Onode sends a message to its neighbors to advertise it has changed its level. It will try to have the same neighbors as the old one using the backup data. Then, it will begin its functionalities as it has been explained in figure 6. Figure 14 shows the algorithm.

When a level-2 Onode fails, it will be detected by its level-1 Onodes neighbors because the lack of keepalive messages for a dead time. First level-1 Onode detects this failure, updates its level-1 Onodes' database and propagates it through the network. When the highest δ level-1 Onode receives this update, it will use the backup information and it will start level-2 Onode functionalities.

Every level-1 Onode has a table with all Dnodes in its area and information related with its level-1 Onode neighbor closest to the level-2 Onode. They will use this table to know their δ and λ. Dnode with highest δ will be the level-1 Onode backup Dnode and it will receive level-1 Onode backup data by incremental updates. This information is used in case of level-1 Onode failure. Level-1 Onode sends keepalive messages to the backup Dnode periodically.

When a level-1 Onode leaves the architecture, it will send a message to its closest level-2 Onode with information about the level-1 Onode backup Dnode.

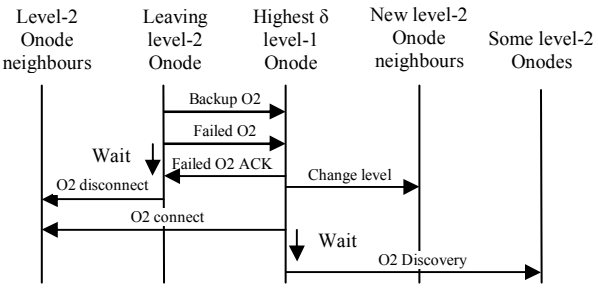

Figure 14. Recovery algorithm when a level-2 Onode leaves.

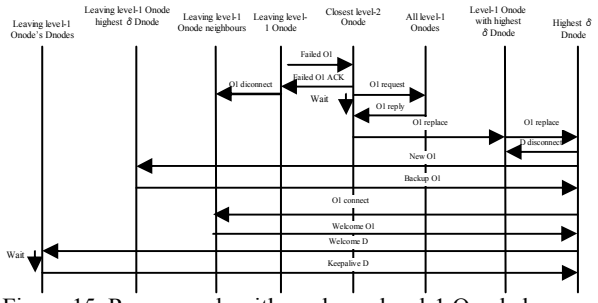

Figure 15. Recovery algorithm when a level-1 Onode leaves.

This level-2 Onode will reply it with the failed O1 ACK message, and then, level-1 Onode will send a disconnection message to its neighbors and leave the architecture. Next, level-2 Onode proceeds as shown in figure 9, but using the received backup data. The level-1 Onode will send a disconnection message to its level-1 Onode, and then, it will connect with the backup Dnode to have the backup data and start level-1 Onode functionalities as shown in figure 10. Then, new level-1 Onode sends a message to Dnodes in its zone. If the level-2 Onode does not receive changes for a hold time, it will send a new request message to the second Dnode with highest δ. If the backup Dnode does not receive this message for a hold time, it will become the new level-1 Onode. Figure 15 shows steps explained.

When a level-1 Onode fails, backup Dnode will check it because the lack of keepalive messages for a dead time. Then, it sends a message to the failed level-1 Onode neighbor. It will be the helper to replace the failed level-1 Onode. Helper level-1 Onode will send a message to its closest level-2 Onode to request a new level-1 Onode. Then, the process will begin as shown in figure 9. When a Dnode leaves the architecture voluntarily, it will send a disconnection message to the level-1 Onode in its zone and to all its adjacent Dnodes from other networks. They will delete this entry from its Dnode's database and adjacent Dnodes will replace it with a new Dnode as explained in figure 8. When a Dnode fails down, level-1 Onode and adjacent Dnodes will ckeck it because they do not receive a keepalive message for a hold time. They will delete this entry from their Dnodes' database and adjacent Dnodes will request a new Dnode for this network.

5. Simulations

We have modeled the architecture as a function of the number of messages sent to the network when a new Dnode joins the architecture and the time needed to converge. Let's consider *n* interconnected logical networks.

5.1. Time to converge

Given figure 8 and taking *tp* as the average propagation time between nodes, we obtain equation 3.

$$T_{D\text{-}D} = (4+d_1+d_2+diameter_3+2 \bullet d_3) \bullet tp \quad (3)$$

We have supposed computer-processing time closer to 0 because it is very low compared with the *tp*. d_1, d_2, d_3 are the number of hops from level-1 Onode to level-2 Onode in the first network, from the first network level-2 Onode to the second network level-2 Onode and from level-2 Onode to level-1 Onode in the second network, respectively, and $diameter_3$ is the second level-1 Onodes network topology diameter.

5.2.1. Study case 1. There are 2 logical networks interconnected (d_2=1). They have one level-2 Onode and several level-1 Onodes. The worst case is when the requesting Dnode is at the edge of the first network (d_1) and the chosen Dnode is at the edge of the second network (d_3). The delay is given by equation 4.

$$T_{D\text{-}D} = (5+d_1+3 \bullet d_2) \bullet tp \quad (4)$$

Let's consider one network with 20 level-1 Onodes and the other with 60 level-1 Onodes and their average number of neighbors is between 2 and 3, so the largest path is 28 hops. We consider 2 cases: (1) Requesting level-1 Onode varies from 1 to 28 hops from the level-2 Onode and destination level-1 Onode is 10 hops from the level-2 Onode. (2) Requesting level-1 Onode is 10 hops from the level-2 Onode and destination level-1 Onode varies from 1 to 28 hops from the level-2 Onode. In figure 12 a), we observe that delay is higher when the number of hops to destination level-1 Onode increases, but it is less significant when the requesting level-1 Onode number of hops increases.

5.2.2. Study case 2. In this case, there are several logical networks interconnected. They have one level-2 Onode and several level-1 Onodes. Requesting and destination Dnode are at the edge of their networks. Both networks diameter are equal ($diameter_1$= $diameter_2$= d). Its delay is shown in equation 5.

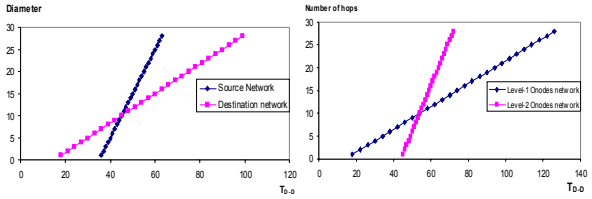

Figure 12 a) & b). Delay to connect 2 Dnodes in cases 1 & 2.

$$T_{D\text{-}D}=(4+d_2+4\bullet d)\bullet tp \quad (5)$$

We consider 2 cases: (1) Distance between networks is fixed to 10 hops and diameter for both networks varies from 1 to 28 hops. (2) Diameter of both networks is fixed to 10 hops and distance between networks varies from 1 to 28 hops. In figure 12 b), we can observe that the delay is higher when the number of hops in the networks increases, but it is less significant when distance between networks increases.

5.2. Number of messages

When a Dnode sends a request to have adjacencies with Dnodes from other networks, it will proceed as it has been explained in figure 8. Let's suppose it needs *d* hops to arrive to the L-2 Onode, so there will be *d* messages. L-2 Onode topology uses the Reverse Path Forwarding algorithm [4], so there will be *n* messages. Let's consider n_j nodes in the *j* network, so there will be n_j messages to request message to know the highest λ Dnodes in that network. Considering 2 highest λ Dnodes at d_{j1} and d_{j2} hops, there will be $d_{j1}+d_{j2}$ messages. Finally, it will proceed as it is in figure 8. All interconnected networks will do this process. Equation 6 gives the number of messages.

$$Messages \Rightarrow 1+d+\cdot n+\sum_{j=1}^{n} n_j+2\cdot\sum_{j=1}^{n}(d_{j1}+d_{j2}) \quad (6)$$

Figure 13 a) shows the number of messages generated when a Dnode requests an adjacency with other Dnodes in the worst case (requesting and destination Dnodes are at the edge), depending on the number of networks, for several number of L-1 Onodes. The worst case is simulated for 86400 nodes (30 nets x 30 L-1 Onodes per net x 96 Dnodes per L-1 Onode). Figure 13 b) shows the number of messages sent through the L-1 Onodes' network over the time when L-1 Onodes have 2, 3, 4 and 5 connections. We observe that the lower number of connections between L-1 Onodes is, higher the time with messages in the network is. However, when the number of connections between L-1 Onodes is higher, the number of messages in the network per time unit increases.

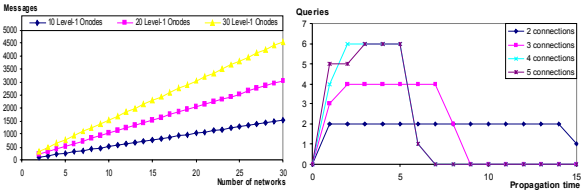

Figure 13 a) & b). Number of messages.

6. Conclusions

We have presented a fault-tolerant architecture, and its protocol, to join logical networks. Their nodes have to share the same type of resources. The protocol organizes Dnodes in zones, allows to join all networks helping to establish connections between Dnodes, and, once Dnodes connections are established, permits to send searches and data transfers between networks. We have defined several parameters to know the best node to promote to higher roles or to connect with. Simulations demonstrate that delay to connect Dnodes is quite higher when the number of hops inside the networks increases than when distance between networks increases. The number of messages in the network when there is a new Dnode, in the worst case, is low compared with the number of nodes in the whole architecture. We have adapted this architecture to CDNs [5] and to sensor networks [6] taking excellent results. Future works will study keepalive time intervals to reduce convergence times. Now, we are developing it for P2P filesharing networks.

7. References

[1] J. Lloret, F. Boronat, C. Palau, M. Esteve, "Two Levels SPF-Based System to Interconnect Partially Decentralized P2P File Sharing Networks", International Conference on Autonomic and Autonomous Systems International Conference on Networking and Services. October 2005.

[2] J. M. McQuillan, I. Richer & E. C. Rosen. "The New Routing Algorithm for the ARPANET". IEEE Trans. on Communications. Vol.28. Pp.711-719. 1980.

[3] C. Cramer, K. Kutzner, and T. Fuhrmann. "Bootstrapping Locality-Aware P2P Networks". The IEEE International Conference on Networks, Vol. 1. Pp. 357-361. 2004.

[4] Yogen K. Dalal y Robert M. Metcalfe, "Reverse path forwarding of broadcast packets", Communications of the ACM. Volume 21, Issue 12. Pp: 1040 – 1048. 1978.

[5] J. Lloret, C. Palau, M. Esteve. "A New Architecture to Structure Connections between Content Delivery Servers Groups". 15th IEEE High Performance Distributed Computing. June, 2006.

[6] J. Lloret, F. J. Sanchez, J. R. Diaz and J. M. Jimenez, "A fault-tolerant protocol for railway control systems". 2nd Conference on Next Generation Internet Design and Engineering. April 2006.