A Self-Adaptable Network Topology for Ambient Intelligence

Boris Mejías, Alfredo Cádiz, Peter Van Roy, Kim Mens
Département d'ingénierie informatique, Université catholique de Louvain – Belgium
{boris.mejias|alfredo.cadiz|peter.vanroy|kim.mens}@uclouvain.be

Abstract

Ambient Intelligence scenarios can be deployed even when the environment lacks of a underlying network infrastructure. This can be done using distributed ad-hoc networks. Ambient Intelligence applications can be highly variable and networks can have an unanticipated number of members. Inappropriate distributed network topologies can lead to unstable and inefficient communication. We propose PALTA*, a decentralized and self-adaptable network topology. We use feedback loops to model its self-adaptable behaviour and we evaluate its performance using different simulations and measurements.* PALTA *allows the construction of distributed networks using self-management techniques and maintaining a good overall performance on the network communication.*

Index Terms: Ambient intelligence, Decentralized systems, Self-management, Network variability

1. Introduction

The Ambient Intelligence (AmI) field envisions people constantly surrounded by hardware technology in the form of interconnected mobile and embedded devices [3]. AmI applications have been proposed for a diverse number of everyday situations ranging from small and close applications, e.g. home and office applications [4, 7], to large and open, such as traffic and massive-events applications [1].

AmI applications can be deployed in any kind of network. Devices can use networks with a defined structure using routers and access points or they could just create they own ad-hoc networks in the absence of such an underlying infrastructure. We are interested in the latter case where the generation and organization of ad-hoc distributed networks is defined by the network members (i.e. peers or nodes) and not by a pre-defined infrastructure.

As mentioned above, AmI scenarios are intended to be deployed in many different situations. In addition, Applications can vary on the functionality given to the users. Since we cannot anticipate the network conditions for each case, we need to define a self-adaptable network capable of optimizing resources depending on the current amount of participants.

We propose PALTA: Peer-to-peer AdaptabLe Topology for Ambient intelligence, a dynamic topology intended for highly variable networks like in AmI scenarios. It combines already developed distributed topologies: *fully connected* networks and the *Relaxed-ring* [6]. PALTA manages the network configuration in order to optimize the communication between peers depending on the network size.

We have submitted a first paper [2] where we provide a detailed motivation for this research and highlight the algorithmic aspects of this approach[1]. In this work, we extend the analysis of PALTA using *feedback loops* in order to study its self-managing properties. Since PALTA has self-adapting facilities for join events and crash recoveries, it is possible to model these features as an automatized system which changes its behavior according to the input it gets from the network state.

We also extend the measurements and analysis of the results obtained from simulations running *fully connected*, *relaxed-ring* and PALTA network topologies. We study the efficient use of resources, the network traffic generated, and the routing efficiency. We discuss each set of results and show the advantages provided by PALTA in the optimization of ad-hoc distributed networks.

Section 2 presents the main concepts behind PALTA. Section 3 describes how PALTA can be seen as a feedback loop. In Section 4 we show the simulations performed over PALTA comparing and discussing the results against the other approaches. We conclude in Section 5.

2. A self-adaptable network topology

In this section we review the main concepts behind PALTA. This is a hybrid topology that optimizes the use of the device's capacity depending of the network conditions. First we present the already existing topologies on which PALTA is based, and afterwards, we present our approach.

[1] This paper have not been published yet, a copy of it can be obtained from the authors of this work.

2.1. Fully Connected topology

When we define a peer-to-peer networks, we can try implementing a direct communication between all the different peers in order to guarantee the network accessibility. These are called *fully connected* networks.

A *fully connected* network means that every peer has an open channel (i.e. a bidirectional link) to every other peer in the network, allowing a direct communication between all the network members. When a new node wants to join the network, it contacts an arbitrary peer sending a *join* message. The contacted peer responds with a *join_ok* message including its full list of connected peers, propagating the *join* message over the rest of the connected peers.

2.2. Relaxed-ring

The *Relaxed-Ring* topology [6] is a Chord-like [8] ring, where every peer has a successor ($succ$) and a predecessor ($pred$). It is a structured overlay network providing a Distributed Hash Table (DHT) where every peer is responsible for a certain range of hash-keys, which is delimited by its own key and the key of its predecessor, $pred$. In order to efficiently route messages in the network, every peer has a set of fingers to jump across the ring. When a new peer joins the network, it uses a hash key as identifier, joining between its corresponding $succ$ and $pred$. In addition, the relaxed-ring allows loosely coupled peers that can be attached in branches when they cannot contact their predecessors. This property makes the system more robust and fault-tolerant.

2.3. PALTA

PALTA maintains a *fully connected* network topology when the amount of peers is smaller than a certain ω value. This is because the network is small enough to handle such a topology. The ω value is defined as a threshold where the fully connected topology can affects the devices and network performance. Therefore, when the ω value is reached, PALTA switches its joining algorithm in order to transform the current network topology into a *Relaxed-ring*. In order to perform a correct transition, PALTA always organizes its peers like a ring, assigning hash-key responsibilities to each one of them. The number of fingers in a fully connected network equals the number of connected peers, but when the network becomes a Relaxed-ring the number of connections per peer decreases, generating lookups requiring more than one hop to reach its destination.

Algorithm 1 shows that the $join$ event in PALTA can be seen almost as a method dispatcher, with the subtlety that it checks its predecessor and predecessor pointers in every join before triggering the $join$ event of the FULL module.

Algorithm 1 Join for PALTA: Adapted fully connected algorithm with transition to relaxed-ring

```
1:  upon event ⟨ join | new ⟩ do
2:      if size(peers) < ω then
3:          check_succ_pred(new)
4:          trigger ⟨ FULL.join | new ⟩
5:      else
6:          trigger ⟨ RING.join | new ⟩
7:      end if
8:  end event

9:  procedure check_succ_pred(id) is
10:     if better_successor(id) then
11:         succ := id
12:     end if
13:     if better_predecessor(id) then
14:         pred := id
15:     end if
16: end procedure
```

In case that ω is already reached, it is the relaxed-ring algorithm in the RING module who will take care of $pred$ and $succ$ accordingly. The procedure that checks predecessor and successor uses functions $better\_successor$ and $better\_predecessor$, to verify whether the key belongs to the range where it is inserted.

The algorithm is written using the event-driven notation. Events are triggered at every node upon the reception of a message. The signature of the event is written between symbols $\langle$ and $\rangle$. What is before $|$ is the name of event. What is after corresponds to the list of arguments.

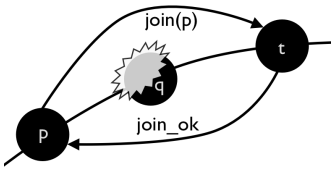

Figure 1. Failure recovery in PALTA.

In order to solve possible inconsistencies due to failures of peers, PALTA uses a *failure recovery* mechanism based on the Relaxed-ring. We can observe the basic action of the failure recovery mechanism in Figure 1. It works as follows: when peer p suspects the failure of its successor q, q is removed from p's routing table. Then, p will search for a new successor in its successor list. Being t the best candidate, p sends a *join* message to t in order to fix the ring. Algorithm 2 shows how this recovery is implemented in PALTA.

The failure recovery relies on a complete and eventually

Algorithm 2 Failure Recovery in PALTA

```
1: upon event ⟨ crash | p ⟩ do
2:     if (p == succ) ∨ (p == succ_cand) then
3:         succ := nil
4:         succ_cand := getFirst(succList)
5:         send ⟨ join | self ⟩ to succ_cand
6:     end if
7: end event
```

accurate failure detector, which is the best you can have in scenarios such as Internet. It means that all crashed peers will be detected and false suspicious will eventually be corrected. Figure 2 depicts a false suspicion of p about a failure of q. In such case, t will refuse the new connection request from p, offering to retry a connection with q. Since the failure detector will eventually provide accurate information about q, the algorithm will eventually converge to a fixed ring. There are more complex approach to converge more efficiently, but that is out of the scope of this paper.

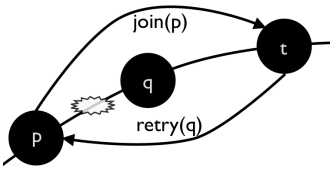

Figure 2. False suspicion of a failure.

3. PALTA as a feedback loop

Now we present the relationship between PALTA and the *feedback loop* technique and how PALTA can be modeled in such systems. Feedback loops are used in systems theory to automatized systems such as air-conditioning or ABS breaks. They can also be found in nature as in our breathing system. Several examples of this can be found in [9], where feedback loops are introduced as a designing model for self-managing software. The loop consists out of three main concurrent components interacting with the subsystem. There is at least one agent in charge of *monitoring* the subsystem, passing the monitored information to a another component in charge of deciding *corrective actions* when needed and an actuating agent is used in order to *perform the corrective actions* in the subsystem. These three components together with the subsystem forms the entire system. It has similar properties to *PID-controllers*, with the difference that the evolution of a running software application is measured discretely.

Let us consider the example of an acclimatized room. Since the goal of the system is to keep the room at a certain desired temperature, the temperature is the value that would be constantly monitored by the loop. This information is given to a thermostat, which is in charge of deciding the corrective action in case the temperature is not the ideal one. If the temperature is too low, the heating system will be activated, behaving as an actuator. When the temperature is too high, it is the air-conditioning is activated.

Following the strategy of [6], where the relaxed-ring is modeled as a feedback loop, we can also model PALTA as shown in Figure 3. The monitors, actuators and the component that decides the corrective actions are placed at every node. The monitored subsystems correspond to the whole peer-to-peer network, and the routing table. The last one is also placed at the node.

As explained in Section 2, when a new node wants to enter the network, it sends a $join$ message to its successor candidate. This message is sent through the network. Since every node is monitoring the network, the $join$ message will be received by the PALTA component. PALTA is also monitoring the load of the routing table. This information is used to decide how to react to the $join$ message. If the load is below ω, PALTA will use the fully connected mechanism together with its own verification of the predecessor and successor. Both actions are used to update the routing table, modifying its load, which will be monitored once again, as in every loop. The fully connected mechanism will also trigger some messages in the peer-to-peer network in order to modify its state. If the load of the routing table has already passed the ω threshold, PALTA will use the relaxed-ring joining mechanism, which will also update the routing table and trigger some messages for the involved nodes.

We can observe some similitude between PALTA's feedback loop and the acclimatized room. The thermostat in the room will use the heating system or the air-conditioner depending on whether the temperature is below or above the desired goal. PALTA decides its actuators according to load of the routing table with respect to ω. In the acclimatized room, the temperature is measure periodically, being trigger by a timer. In PALTA, it is the join message the one that triggers the monitoring process and the rest of the loop.

The loop also monitors failures of peers triggering the corresponding failure recovery mechanism. This mechanism is chosen by PALTA according to the load of the routing table, as it is done with the joining process. This is coherent with what is explained in Section 2. All other messages related to the joining process and the failure recovery, such as $join\_ok$, new_succ, are also present as monitoring event, but they have been omitted from Figure 3 for legibility.

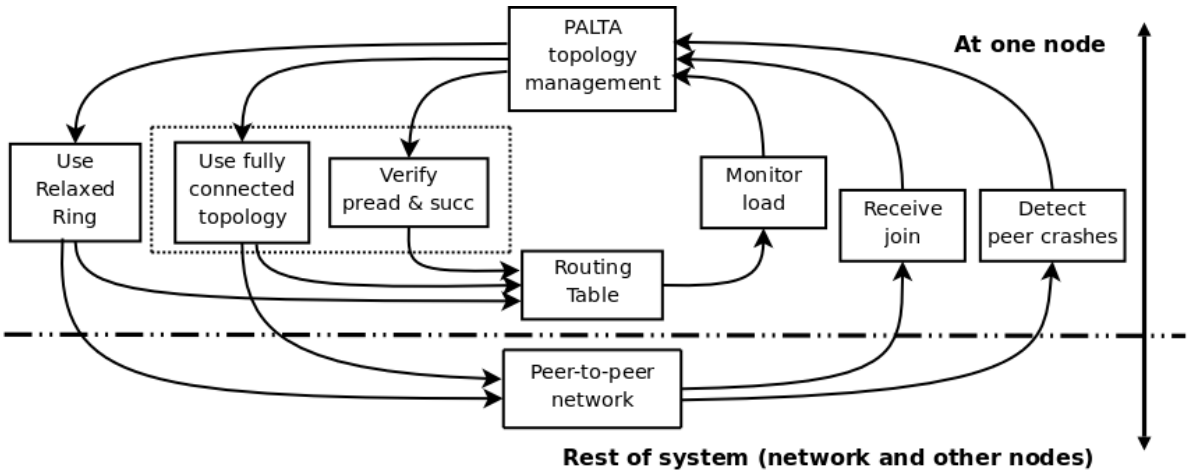

Figure 3. Self-Adaptable topology as a feedback loop

4. Validation and Discussion

This section presents an analysis of the results obtained by simulating PALTA, the relaxed-ring and a fully connected network. The simulation is implemented using CiNiSMO [5], where the code of every node runs in its own light-weight thread. Peers communicate with each other by message passing using ports. In order to measure the efficient use of resources, we have measured the average amount of active connection a node has in every of these networks. To study the performance of the topologies, we have measured the total amount of messages needed to build the network, and the average hops needed to perform a lookup.

Every topology is tested by building networks from 20 to 1000 nodes, increasing the size by 20 nodes at every iteration. Plotted values represent the average of running every experiment with several seeds for random number generation. In the case of PALTA, we tested the algorithm using two different values for ω, being 100 and 200. Reaching 1000 nodes might be considered not large enough for large scale networks, but it is enough to observe the behavior after the ω threshold is reached and extrapolate the scalability from the curves obtained.

4.1. Active connections

One of the goals of PALTA is to dynamically adapt its topology in order to optimize the use of the network. For small networks that means that we want to directly connect as much peers as possible, in order to reach every peer in the minimum amount of hops. Small is defined in terms of the ω value.

Figure 4 shows the average amount of active connections per peer in the different topologies. We can observe that the *fully connected* network increments the amount of connections linearly, and therefore, it does not scale at all. Part of the curve is missing, but it clearly corresponds to $n - 1$, being n the size of the network, because every node is connected to all the other peers. As expected, the *relaxed-ring* appears as the topology where peers manage the smallest amount of connections, showing that it has good scalability for large networks. Let us analyze now the behavior of PALTA. In both cases, with ω 100 and 200, we observe that the amount of connections increases linearly as a fully connected network until reaching ω peers. From that point on, the average of connections decreases very fast, converging asymptotically to the values of the relaxed-ring. This is because all new nodes that join the network after the threshold of ω is reached, create only the amount of fingers needed by a relaxed-ring. In fact, ω peers manage $\omega - 1$ connections, and $N - \omega$ peers manage k fingers, with N being the size of the network. Meaning that the larger the network, the smallest the average. Of course, this decreasing behavior continuous until it almost reaches the curve of the relaxed-ring, then, the average can only increasing according to the size of the network.

In conclusion, Figure 4 shows us that PALTA uses actively more resources than a regular ring, but it is capable of self-adapting when the network becomes too large and provide a good scalability.

4.2. Network traffic

When peers enter in a distributed network, they generate a number of messages in order to correctly join without leading the network to an unstable state. In the case of a *fully connected* network, the joining peer will always need $2 * n$ messages to contact all peers in a network of size n. Therefore, the cost of a new joining peer increases as the

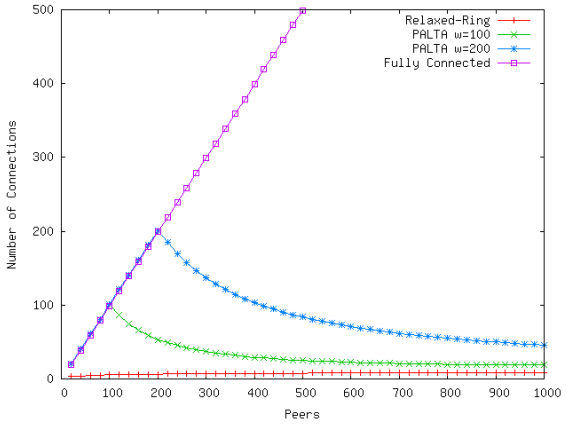

Figure 4. Average amount of active connections vs number of peers.

size of the network increases. In our simulation we contact directly every peer. In case a broadcast mechanism is used to propagate the *join* of a new peer, n messages are needed to reach every peer, plus n message to acknowledge the new peer, making $2 * n$ messages.

In a *relaxed-ring*, the joining peer needs to send messages for contacting the predecessor, successor and the k fingers. Therefore, the marginal cost of a joining peer is almost independent of the size of the network. The only difference occurs with the amount of messages needed for localizing the k finger, which increases logarithmically with respect to the size of the network, as we will see in Section 4.3.

Figure 5 does not show the marginal cost of joining a network, but the total amount of messages generated to construct every network we have studied in section 4.1. We can see that with less active connections, as in PALTA or the relaxed-ring, the number of messages remains small, generating less network traffic. The curve of the fully connected network increases quadratically, generating $n*(n-1)$ messages, with n being the size of the network, we can conclude that this network cannot scale.

The curve of the relaxed-ring shows a constant and controlled increment in the amount of messages, keeping them at a very low rate, showing that it scales very well. Now, the results obtained from experiments with PALTA are very interesting because both perform better than the ring for larger networks. One can observe that PALTA with $\omega = 100$ and $\omega = 200$ increases quadratically the amount of messages, as in a fully connected network. This happens only until the network reaches a size of ω peers. Then, the amount of messages increases slower that in a ring, and furthermore, after a certain size of the network, both PALTA networks remain at better values that the relaxed-ring. The explanation for this is that when a new peer join in the network, it needs less messages to find the k fingers. This is because PALTA has ω peers with a larger routing table ($\omega > k$), making a more efficient jump during the routing process. We study this further in the following section.

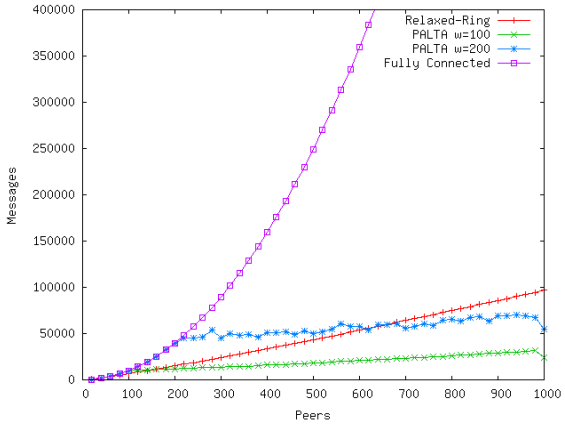

Figure 5. Total amount of messages to build the network vs number of peers.

This means that the cost of maintaining a small fully connected network can help a larger network to be more efficient for routing, generating less network traffic. We observe that PALTA could not only be used for ambient intelligent networks, but also as the topology for large scale systems.

4.3. Hops

In order to confirm our conclusions from the previous experiment, we decided to measure the average amount of hops needed for a message to reach its destination. This is known as a *lookup operation* in a ring. This experiment does not consider fully connected networks, because there is no concept of responsibility is such systems. In addition, because of its characteristics, peers in a fully connected network reach any other peer in the network in 1 hop.

In Figure 6 we can observe the results obtained. The relaxed-ring shows that the number of needed hops increase logarithmically when the network size increases. PALTA performs better than the relaxed-ring due to fact that some peers have a larger routing table, confirming the results from the previous experiment. In both cases, PALTA presents an average number of hops slightly smaller than 2 if the network consist of less than ω peers. This is because the network is fully connected, and therefore, in can reach the predecessor of the responsible of the looked up key in only one hop. The second hop is needed to reach the responsible. The average is smaller than 2 because the randomized experiments sometimes generates lookups where the responsible is the peer triggering the lookup.

After the value of ω is reached, the average increases faster in PALTA with $\omega = 100$ that with $\omega = 200$. This is clearly due to the amount of peers having a larger routing table. We observe that in both cases the system behaves much better than the ring. We expect that for larger networks the value would converge to the curve of the ring, but still performing better. What we cannot currently explain is the behavior of PALTA with $\omega = 100$ when the network is in between 100 and 200 nodes. It seems to perform even better than a $\omega = 200$.

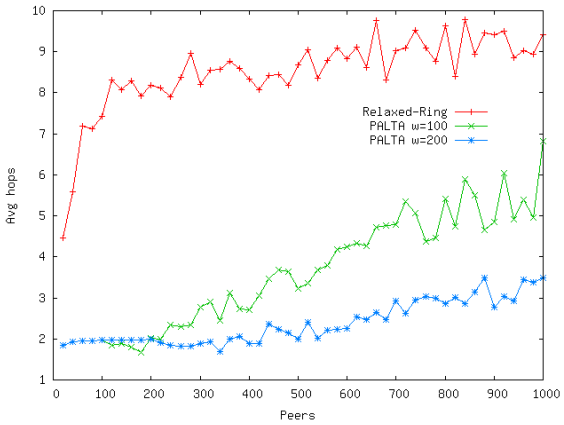

Figure 6. Average number of hops vs number of peers.

Something that we still need to investigate is the construction of a network where every peer define its own ω vale according to its own resources. We want to PALTA in ambient intelligent network formed by heterogeneous devices, each one with its own resources.

5. Conclusions

Ambient Intelligence envisions the ubiquitous presence and cooperation of computers in order to enhance our quality of life. Mobile devices are helping us to achieve this goal because they are becoming more powerful and they have better communication capabilities. We are interested in how mobile devices can organize themselves into ad-hoc networks when the environment lacks of a network infrastructure. We present PALTA a self-organized and self-managed network topology which can optimize the use of the network and device resources depending on the network conditions.

In this paper we have presented the main ideas behind PALTA. We study its self-managing properties by using feedback loops to model its behaviour. We also presented and analyzed the results obtained from simulations performed with different network configurations and the different topologies we studied. PALTA shows good results compared to other approaches: it optimizes the use of the resources in different network configurations, providing a better *use of the resources* in small networks, but when networks become bigger it can adapt its organization for providing better *scalability* without modifing the connections links of the already connected peers.

We believe that this approach is useful for the self-organization of mobile devices in highly variable environments.

6. Acknowledgments

This work has been supported by projects VariBru, SELFMAN and MoVES. The authors would like to thank S. González and P. Hass for their comments on this work.

References

[1] G. Cabri, L. Ferrari, L. Leonardi, and F. Zambonelli. The laica project: Supporting ambient intelligence via agents and ad-hoc middleware. In *WETICE '05: Proceedings of the 14th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprise*, pages 39–46, Washington, DC, USA, 2005. IEEE Computer Society.

[2] A. Cádiz, B. Mejías, J. Vallejos, K. Mens, P. Van Roy, and W. De Meuter. PALTA: Peer-to-peer AdaptabLe Topology for Ambient intelligence. In *SCCC '08: XXVII International Conference of the Chilean Society of Computer Science*.

[3] K. Ducatel, M. Bogdanowicz, F. Scapolo, J. Leijten, and J.-C. Burgelman. Scenarios for ambient intelligence in 2010. Technical report, EC Information Scociety Technologies Advisory Group (ISTAG), 2001.

[4] C. D. Kidd, R. Orr, G. D. Abowd, C. G. Atkeson, I. A. Essa, B. MacIntyre, E. D. Mynatt, T. Starner, and W. Newstetter. The aware home: A living laboratory for ubiquitous computing research. In *CoBuild '99: Proceedings of the Second International Workshop on Cooperative Buildings, Integrating Information, Organization, and Architecture*, pages 191–198, London, UK, 1999. Springer-Verlag.

[5] B. Mejías. CiNiSMO: Concurrent Network Simulator in Mozart-Oz. *http://p2ps.info.ucl.ac.be/cinismo*, 2008.

[6] B. Mejías and P. Van Roy. A relaxed-ring for self-organising and fault-tolerant peer-to-peer networks. In *SCCC '07: Proceedings of the XXVI International Conference of the Chilean Society of Computer Science*, pages 13–22. IEEE CS, 2007.

[7] N. Sadeh, F. Gandon, and O. B. Kwon. Ambient intelligence: The mycampus experience. Technical Report CMU-ISRI-05-123, School of Computer Science, Carnegie Mellon University, July 2005.

[8] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan. Chord: A scalable Peer-To-Peer lookup service for internet applications. In *Proceedings of the 2001 ACM SIGCOMM Conference*, pages 149–160, 2001.

[9] P. Van Roy. Self management and the future of software design. In *Formal Aspects of Component Software (FACS '06)*, September 2006.