TOWARDS DESIGN OF AN OVERLAY ARCHITECTURE IN THE MULTINETWORK MANAGEMENT SYSTEM

Zhijing Qin, University of California, Irvine
Luca Iannario, University of Bologna, Italy

CONTENTS

1. Background and motivation 1
2. Related work and high level design 4
  Communication and computation locations: 6
  Network selections: 7
  Convergecast based data collection: 9
  Routing mechanisms: 11
  Wireless sensor network management: 13
3.High level design thoughts: 15
  High level Design: 15
    Server architecture 18
    Client architecture 21
4. Overlay construction problem formulation and objectives: 24
  Algorithms: 25
    Data structures: 25
    Initialization: 25
    Maintenance: 26
  State Transition Graph: 27
  Pseudo Code: 29

1. BACKGROUND AND MOTIVATION

The emerging advance of devices in pervasive computing such as sensors and smartphones have made it feasible to install various kinds of wireless technologies in these devices to either distribute or request services for a variety of purposes. For example, fire sensors can use ZibBee to report the temperature values and smoke density to firefighters in real time; smartphone users can request video services either from cellular network or Wireless LAN; music fans can enjoy amazing albums via Bluetooth earphone speakers. However, with the increase of the user number and varieties of applications, the shortcomings of each wireless network begin to emerge. Cellular networks have issues when serving a large volume of clients. In some urban areas, dropped calls can reach 30% [1]. And cellular network cannot handle large scale live video distributions since existing cellular deployment do not natively support broadcast and multicast[2]. Actually once a measurement study shows that each UMTS HSDAP cell can only support 4-6 mobile video users at 256kbps [3]. Even in 3G and 4G systems, such as Long Term Evolution (LTE) and WiMax, the data rate they provide is also shared among all the users of a base station. When the population inside a cell becomes bigger, they will face severe scalability problems.

WiFi networks can significantly help scale cellular networks, especially for large volume data distribution [2]. However, WiFi interface on mobile devices consumes high energy even in Power Saving Mode [4]. In [1], the author's experiments show that the energy consumption of wifi card in idle and active mode is 29.42 mw and 1648.2 mw respectively. While the consumption is 2.57 and 94.5 for Zigbee, 7.32 and 340.3 for Bluetooth. So for small traffic load application, zigbee and Bluetooth are the better choice.

Zigbee network can be used to transmit data as rates vary from 20 – 250 kbps while consume quite low energy. It can be used individually in home and industrial control, medicine system, and also it can be cooperatively used with WiFi to send voice data as low energy consumption. Due to the low link bandwidth, Zigbee cannot handle large volume data like video and file sharing.

Bluetooth is a short-range wireless technology aimed at replacing cables that connect phones, laptops, and other portable devices. Transmission rate varies from 1Mbps – 2Mbps, which is slightly higher than Zigbee. But Bluetooth networks or "piconets" support up to only eight devices communicating simultaneously. If the network size scales up, there must be more piconets forming up a scatternet, wgucg will leads to increased communication latency. Another shortcoming of Bluetooth is its perioadcal waking up and synchronization with the master device of the piconet. A Bluetooth device may consume approximately 3s to wake up prior to synchronization [5].

Besides specific wireless technology, network infrastructure also plays an important role on reliability issues. Most of the wireless technologies mentioned above rely on the static infrastructures, e.g. cellular tower, wireless access point. Once these infrastructures failed, the whole network will be down. Hence to improve network reliability, we propose to use multiple network access modes including infrastructure based and ad hoc based communication.

We believe that combining multiple wireless technologies and infrastructures can greatly improve the network performance in terms of connectivity, bandwidth, latency, reliability and energy efficiency (Figure 1 shows a simple scenario of the Multinetwork environment). To facilitate this combination, we proposed a multinetwork management system here. As shown in figure 2, wireless technologies have different characteristics such as data rate and transmission range. Similarly, devices also have different features like power and computational capabilities. So there should be a general mechanism to coordinate such a heterogeneous environment. Second, the network administrator needs a common image of the Multinetwork to make it better used. For example, some hot spots or bottlenecks of the network should be discovered and network agents will be deployed there to improve the traffic throughput capability. Third, besides the opportunities like reliability and connectivity enhancement that the Multinetwork brings, it also produces some challenges. For an instance, multiple wireless radio technologies will interfere with each other and decrease the performance of the entire network. The basic idea is that first collect the network state information from nodes and links, then analyze the collected data to provide support for either end to end application or network administration tasks. However it is not an easy task. There are three main challenges:

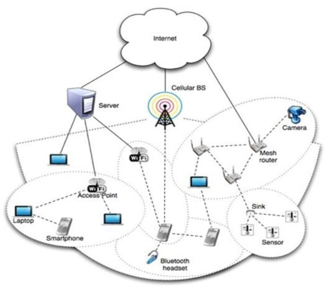


Figure 1

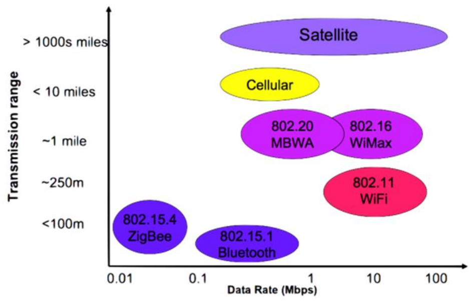


Figure 2

1. Organize heterogeneous nodes. Some of them are mobile and battery supplied and some of them are stationary and plugin powered. We should consider both scalability and dynamicity in such a multinetwork environment.

2. Design and implement a network state information model. This information model should have a general format for all wireless technologies and nodes with different capabilities, and can be extended specifically if necessary. Also it should have several features: efficient storage, representation and transmission.

3. Effective and efficient tools. When network state information ready, there should be an effective and efficient analyzing and reasoning tools to give proper supports to those nodes that have multiple communication opportunities.

Dealing with a Multinetwork management system is challenging for different reasons but, at the same time, it is very useful in different scenarios and for different kind of users. In the following paragraphs we are describing some meaningful use cases for our system.

If we think about a human network administrator, for instance, it is useful to have an up-to-date big picture of the Multinetwork global state. The global state can be defined as the status of the devices that are participating in the managed network. In this way, the administrator can monitor the current status of each managed device and can solve transient faulty behaviors. Considering for instance a campus network, it is possible to analyze the current global state and detect possible faults, bottlenecks, congestions and unexpected behaviors. To avoid bottlenecks, for example, it is possible to balance the traffic load in between the different networks, considering application requirements and networks availability.

Another relevant use case for our Multinetwork management system is related to disaster recovery, for example after earthquakes, hurricanes, etc. After a significant disaster, indeed, it is really likely to have some service disruptions due to major physical damages. Thus, in this scenario it is really useful to know the current state of the network in order to identify possible faults and drive access selection to correctly route on-field communications. Let us stress how it is important to have a quick recovery after a major disaster. Our system can help to efficiently route traffic among the active networks, enabling rescuers to share information (for instance) about people in danger.

It is possible to identify a third use case from the end-user perspective as well. Customers are always more interested in QoS-aware applications and service providers need to fulfill application requirements as much as possible. Considering the actual network load (e.g., bandwidth utilization), application requirements, and the nearby available networks, for instance, it is possible to give some “hints” to the user device in order to switch to the best connectivity opportunity available at that time (or in the near future, if the user is moving and we can predict its trajectory).

For now we mean management by collection and analysis of the network state information, and do not perform many operations to manipulate the devices. For those devices that are hard to manipulate, such as cellular base station, we will only try to collect information like signal strength, rather than touching the inside of these devices. To sum up, a Multinetwork management system is needed in presence of coexisting multiple networks, in order to better utilize the network without many side effects.

The first (and also the basic) task in Multinetwork management system is to construct an overlay network upon physical nodes and links. Basically we classify the nodes in Multinetwork into three categories: central database, stationary nodes and mobile nodes. Central database issues collection commands into the network and stores the network state information collected from other nodes. It resides in the Tier 1 and has unlimited power supply and strongest computation capabilities. Stationary nodes such as access points, mesh routers, and cellular base stations, have wired connection with the central database. They collect network information from other nodes in the network and report them to the central database. They reside in the Tier 2 and have unlimited power supply and strong computation capabilities. Mobile nodes like sensors, motes, laptops, and smartphones can be either associated with nodes in Tier 2, or formed as a cluster by themselves. They receive the collection commands and send their state information towards the central database. They reside in Tier 3 and have limited power supply and low computation capabilities. Figure 3 gives an example of our proposed Tier-based architecture.

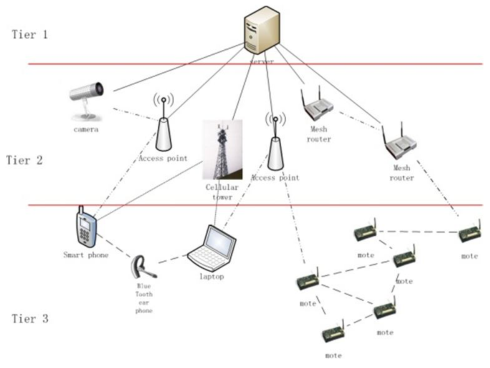


Figure 3

2. RELATED WORK AND HIGH LEVEL DESIGN

For state collection in multiple network environments, we argue that there are two main issues that should be addressed concurrently: one is how to deal with the heterogeneous wireless access opportunities; the other one is how to organize the nodes considering the traffic pattern in data collection. For the first issue, we take [1] as a main reference, which surveys a substantial number of literatures on heterogeneous network. We choose related works that may give inspirations on our work and proposed a different classification upon them to facilitate the full exploitation in Multinetwork management context. For the second issue, the convergecast transmission behavior in data collection forces us to give a concrete survey in tree based convergecast algorithm in WSNs. At the same time, given the difference between our Multinetwork management environment and the traditional wireless sensor networks, such as mobility and dynamicity, we would also like to investigate routing mechanisms in MANET, which can be integrated with convergecast algorithm to be better applied in our Multinetwork management scenario.

Bellavista et al. [6] gave a unifying perspective on Context-aware Evaluation and Management of Heterogeneous wireless connectivity. They indicate the integrated management on multiple networks, with full context visibility, of all connectivity options available at runtime as Context-aware Autonomic Management of Preferred network Opportunities (CAMPO). They also proposed a model to descript such kind of systems in terms of relationships between three types of entities, via reviewing a bunch of works on heterogeneous wireless network connectivity and management:

- Application represents a running service client at a terminal. It actively requests connectivity to fulfill its application goals.
- Interfaces model the wireless hardware equipment available at clients, e.g. Wi-Fi and Bluetooth cards.
- Connectors are the entities actually providing client nodes with connectivity by interworking with client-side active interfaces, including Access points and wireless peer connectors.

In the simplest scenario, only one interface is usually active at a given time, even if several of them are potentially available for clients. They use channel triple <N:1:1> to indicate that the interface selector relationship is N-to-1 and the connector selector relationship is 1-to-1. In the more complicated scenario, multiple interfaces may be simultaneously active and the main support goal is to activate and update the most suitable channels for any running application. The motivation is that different applications usually have different service-specific requirements in terms of bandwidth, latency, and sustainable discontinuity intervals. Thus the possibility to have channels with different interfaces for different applications at the same time can significantly improve the exploitation of available connectivity opportunities. In this more complicated scenario, there are two main categories of the solution depending on the cardinality of the connector selector relationship. On the one hand, each activated interface associates with a single connector (<N:M:M>). One the other hand, some solutions additionally consider the possibility of associating multiple connectors with each active interface, thus enabling applications that exploit the same interface but different connectors (<N:M:L>).

Based on this model, the authors gave several categories in different levels and put a bunch of related work on heterogeneous network into these categories.

However, for our Multinetwork management system, we would like to give our own different classification, although motivated by [6], with the following reasons:

- They mainly focus on the hybrid networks between cellular and WLANs. In our case, we would like to propose a management system focused mainly on unlicensed radio network, i.e. Wi-Fi, Bluetooth, Zigbee, etc. Even if we do not expect we can modify the core part inside a cellular tower, we can still get some information about a cellular network from a client view such as signal strength.
- Most of the systems appeared in [6] are communications system, rather than management system. They usually provide management functionalities when interfaces/connectors selection happened during a communication process. And the application traffics may vary from multimedia streaming to file sharing. In our case, the data transmitted is related to network state information and there is a central database receiving and storing that information.

- Some of the classifications in [6] are not suitable for a management system, such as AAA or billing information.
- They did not pay attentions on routing mechanisms since the IP routing functionalities are used by default. Since there are more mobile nodes and possibilities of ad hoc networking in our scenario, it is necessary to consider routing protocols either in application layer or network layer.

We examine the literatures not only referred in [6], but also from a more wide perspective in terms of network management. Here we would like to organize the literatures into three main categories, which can better facilitate the design of overlay architecture in the Multinetwork management:

COMMUNICATION AND COMPUTATION LOCATIONS: it means where the connectors and computations tasks are located. For the connector location, it is either located at infrastructure side and peer side. [7] monitors the performance indicators about the current used network and of other current connectivity opportunities (WLAN and cellular networks). It uses 802.11e to estimate the WLAN bandwidth and perform vertical handoff between WLAN and cellular network according to the WLAN estimated bandwidth. The exploitation of peer connectors requires additional capabilities in the deployment environment and introduces further complexity in context evaluation. [8] proposed a two hop relay architecture, from mobile nodes (MN) to relay gateway (RG) to cellular network. When the WLAN connectivity is not available, the MN will explicitly request for RG-based connectivity. In [9] the proxy client can interwork with both cellular and wireless ad hoc network. MN can interact with proxy client not only directly, but also via intermediate peer connectors. If the downlink from a cellular network to a MN is not good, the content can be sent first to a proxy client and then forwarded to the proper destination. CHUM [10] dynamically elects one node to play the role of gateway between the MANET and the fixed network infrastructure. For the computation location, it is either client transparent (located at server), or end-to-end mode (located both on client and server), or proxy mode. [11] eliminates the need for direct service level agreement by using a third party software—network interoperating agent (NIA), which is running in the fixed Internet and interworking gateways (IGs) residing on the integrated and heterogeneous wireless networks. The centralized NIA provides interoperability capabilities between IGs, thus eliminating the need of direct service level agreements between each pair of involved networks. Connectivity middleware management in [12] enable adaptive connectivity by working primarily on clients. In particular, it offers functions for event-based monitoring of available interfaces and for commanding channel switching. PROTON [13] deploys a formal policy representation model, based on finite-state transducers, that evaluates policies using information from the context to manage mobiles' behavior in a transparent manner, hiding 4G systems' complexities. Different from the above solutions, [14] are based on component deployment on the infrastructure side. It predefined some congestion scenarios in data base. Every time there is congestion happened in the network, it will try to match it with the predefined scenario, and if match found, strategies will be applied; otherwise, global management will transfer some traffics from the congestion network to others.

By summarizing this part, most proposals exploit infrastructure-based connectors since they mainly focus on infrastructure networks such as WLAN and cellular network. However, given the emerging peer-based networks such as MANET and WPAN, and increasing node capabilities, we need to consider new scenarios where client terminals are also exploited to offer connectivity opportunities and thus the peer-located communication solutions are gaining relevance.

For computation location part, there are three main reasons that let us argue that client side solutions should get more attentions: a) the growth in node capabilities coupled with the proliferation of heterogeneous wireless networks is pushing for privileging client-side systems, mainly to facilitate development and deployment over open wireless environments. b) Deploying more computation tasks in client side can greatly increase the scalability of the entire system. c) Most of the management system cannot touch the core part of the cellular network; it is more valuable to devise approach to monitor the network condition from the perspective of users/clients. However, purely relying on client computation capabilities also has some limitations: a) some battery-supplied nodes may not have enough computation ability; b) local sub-optimal solution may not be close to the global optimal solution. Hereby we argue that a hybrid location combined client-based solution and proxy-based solution should be proposed. On one hand, distributed nodes can exploit the cooperation with their local neighbors and, on the other hand, the proxy should provide extra information and coordinate these nodes to achieve better performance.

NETWORK SELECTIONS: the goal of the network selection mechanisms is to quantitatively measure, in a homogeneous and comparable way, the current suitability of possibly heterogeneous interface-connectors to be taken into account for overlay network formation. The network selection is necessary both at the system bootstrap time and to update active connections at provisioning time, which corresponds to overlay construction and maintenance respectively. We divide the network selections into three phases: input collection, selection process, and output. The general workflow is that the system first collects some related information either from users or devices, then makes decisions via predefined or dynamically adapted functions, finally choose the output interface which the outgoing data should be sent from.

Simple network selection solutions exploit a static priority order among available interfaces; the only dynamic input data to consider is network availability, often based on beacon frames [15]. More recent solutions exploit more dynamic input information to make the selection more reasonable. Some contributions focus on a small set of physical level network parameters. [16] examines interface power consumption in transmit/receive/idle state and download/upload link (by checking the length of input and output queue). Similarly, [17]is also based on physical input data, i.e. RSSI of APs and interface embedded RSSI thresholds for handoff triggering. Besides, the handoff threshold values can vary dynamically, depending on mobile node speed and on handoff failure probability requirements. [7] dynamically evaluates performance indicators for both WLAN APs and cellular network base stations. In the case of WLAN, it monitors RSSI variations (physical layer) and residual bandwidth (network layer), which are derived from direct measurements of throughput, channel utilization, and frame loss rate (the last two indicators are available in QoS BSS beacon frames). In the case of cellular networks, it exploits statically defined nominal values. [18], instead, applies the Fast Fourier Transform to RSSI values of Aps in proximity to quickly and accurately detect signal decay. In addition, it exploits network-level information: the Network Allocation Vector provided by IEEE 802.11 APs is used to infer bandwidth and access delay. In that way, it can select the less loaded AP in a set of eligible ones with RSSI over a threshold. [19] considers as input data user profiles (subscribed services, corresponding QoS requirements, and maximum price allowed), terminal profiles (client device hard-ware/software capabilities), network offers (currently available services, supported QoS levels, and corresponding costs), and configuration costs (time and, more generally, resources required to perform channel reconfiguration). Among these parameters, user/terminal profiles are rather static, network offers and configuration costs may be very dynamic indicators. By summarizing, the relevance of exploiting dynamic input from each layer including context input to evaluate the connectors has rapidly emerged and now is widely recognized. But the issue of efficiently retrieving context information at a high level of abstraction and how to properly use it is not fully addressed.

The approach of the selection processing remarkably influences the effectiveness and efficiency of communication over heterogeneous networks. The most two important aspects mentioned in [6] are flexibility and objectives. About flexibility, embedded processing methods permit to define how to combine input data only before the starting of service sessions. Extensible methods, instead, are modifiable also during service provisioning, either function based or policy based. Function based metrics typically calculate output as a linear function of input and weights, which may be adaptively configured also at runtime. To further increase flexibility, some metrics are defined depending on high-level declarative obligation policies. About objectives, in the case of local scope, the processing methods only consider the local requirements such as minimize local power consumption; in the case of global scope, the processing method aims to achieve a network wide goal, such as balancing network load. [20] has an embedded method that makes handoff between WLAN and UMTS depending on whether the RSSI of APs overcome or fall below a threshold. Some solutions employed more complicate algorithm to process the input information and provide support to the selection decision, such as dynamic programming [21], pattern recognition [22], neural network [23] and fuzzy logic [24]. In [25] Users can select their personalized best network by changing weight factors and constraints in a single objective optimization problem. [26] provides users with the capability to specify a priority order among network characteristics, by defining a proper weight set. Then it exploits the weight set to evaluate a linear combination of network conditions, network performance, service cost, power requirements, security, proactive handoff, and client speed for each active interface. Other proposals exploit more processing methods, for instance based on the knapsack algorithm [27] and on the Analytic Hierarchy Process [28]. In [27], each traffic flow is modeled as the set of its associated bandwidth/delay requirements and a partitionability flag; any available network is represented by its maximum bandwidth, maximum delay, and power consumption indicators. [28] exploits AHP to decide weights and Grey Relational Analysis (GRA) to rank channel alternatives. AHP splits a complex problem, such as the provisioning of the best QoS, into a number of decision factors: availability (decomposed in RSSI and coverage area), throughput, timeliness (delay, response time, and jitter), reliability (BER, burst error, and average retransmissions per packet), security, and cost. On the one hand, GRA normalizes and compares UMTS and WLAN QoS parameters; on the other hand, it exploits AHP to determine Grey Relational Coefficients and thus to choose the most suitable interface. To further improve flexibility and extensibility, some network selection solutions adopt policy-based methods. [13] exploits policies as event-condition-action rules, i.e. declarative rules that specify actions to execute whether conditions apply, with events that trigger condition evaluation. Context input data are not considered aggregately, as in many function-based metrics: PROTON breaks down context into fragments and allows the specification of independent normalization for any fragment. In particular, PROTON permits to define tautness functions to determine how tautly a condition fits to an event: the closer the returned value to 0, the tauter the condition to a specific event. [29] performs policy decision and enforcement on both client and infrastructure-side; the primary overall objective is to balance networking load among overlapping cellular and WLAN networks.

We noticed that [27] and [29] target a global objective: e.g. monitoring the performance of each considered network and optimally distributing network traffic load. Others target a local objective.

In conclusion of this subsection, we found that embedded process methods share the common non-negligible limitation of not allowing to change processing method requirements at runtime. Most recent systems are proposing function-based solutions, which aim to achieve the optimal trade-off between flexibility and computational complexity. There are also some solutions adopting policy based processing methods that provide greater flexibility but omit to propose effective metrics in terms of imposed overhead. Thus function based process methods will continue to be adopted in the next few years.

Network selection can be considered as a local operation to achieve some goals such as load balancing, energy efficiency and high data rate. However, given a network usually we would like to achieve these goals by considering the route from an end-to-end perspective. In the following, we first examine the related work on convergecast, which is a common communication pattern in wireless sensor network, and then we will give a survey on routing mechanisms in MANET, considering mobility and dynamicity which is usually omitted in wireless sensor network.

CONVERGECAST BASED DATA COLLECTION: Data collection from a set of sensors to a common sink over a tree-based routing topology is a fundamental traffic pattern in wireless sensor networks (WSNs), which can be brought into network state information collection domain. This many-to-one communication pattern, in which data flows from many nodes to a single node, is known as convergecast. One may view convergecast as opposite to broadcast or multicast in which data flows from a single node to a set of nodes in the network. Figure 4 shows a simple example that illustrates the characteristics of a typical broadcast and convergecast. In broadcast, as shown in Figure 4a, node s is the message source and nodes a, b, and c are expected recipients. Node a hears the message directly from s and forwards a copy to nodes b and c. In case of a convergecast, as shown in Figure 4b, nodes a, b, and c each has a message destined to the s ink node s and a serves as a relay for b and c.

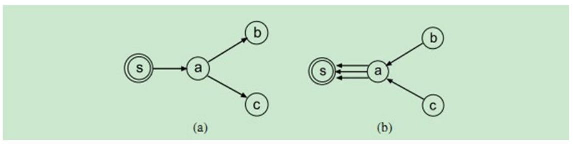


Figure 4

Particularly under regular, heavy traffic conditions, contention-free medium access control (MAC) protocols, such as time division multiple access (TDMA), where nodes communicate on different time s lots to prevent conflicts, offer several advantages for data collection as compared to contention-based protocols. They eliminate collisions, overhearing, and idle listening, which are the main sources of energy consumption in wireless communications. In addition, they also permit nodes to enter into sleep modes during inactive periods, thus achieving low duty cycles and conserving energy. Furthermore, TDMA-based communications can pro-vide provable guarantee on the completion time of data collection, for instance, in timely detection of events. Another key aspect of time-slotted communication is robustness during peak loads. When source nodes are many or the data rates are high, carrier-sense multiple access protocols, such as CSMA, mayfail to successfully allocate the medium, causing retransmissions and collisions.

Unlike raw-data convergecast where the application requires every single packet generated by the nodes to be delivered to the sink, periodic data collection often requires delivery of only summarized information in the form of aggregated packets. In general, such aggregated convergecast requires less number of time slots than raw-data convergecast because of the reduced volume of traffic enrooted to the sink. Under this setting, it is assumed that every node generates a single packet at the beginning of every frame and perfect data aggregation is possible, i.e., each node is capable of aggregating all the packets received from its children as well as that generated by itself into a single packet before transmitting to its parent. This means that the size of aggregated data is constant and does not depend on the actual raw sensor readings. Typical examples of such aggregation functions are MIN, MAX, MEDIAN, COUNT, AVERAGE, etc., which are known as algebraic and distributive functions.

There are three main objectives in continuous aggregated data collection: minimize TDMA schedule length, minimize data latency and minimize energy consumption. To minimize the schedule length, each parent node ideally should wait to receive all data from its children and then aggregate those with its own data before transmitting. However, a special case that a parent node need not wait to receive all data from its children within a single frame before transmitting is investigated by Incel et al. [30] and Ghosh et al. [31]. This is particularly applicable for continuous and periodic monitoring applications that sustain over long durations of time, such as network state monitoring. The authors in [30] explore a number of different techniques that provide a hierarchy of successive improvements, the simplest among which is an interference-aware, minimum-length TDMA scheduling that enables spatial reuse. To achieve further improvement, they combine transmission power control with scheduling and use multiple frequency channels to enable more concurrent transmissions. The multiple frequencies are assumed to be orthogonal, and a receiver-based channel assignment (RBCA) scheme i s proposed where the receivers (i.e., parents) in the tree are statically assigned different frequencies to eliminate interference. It is shown through extensive simulations that once multiple frequencies are used along with spatial-reuse TDMA, the data collection r ate often no longer remains limited by interference, but by the topology of the network. Thus, in the final step, degree-constrained trees are constructed that further enhances the data collection rate. Ghosh et al. in [31] prove that minimizing the schedule length under multiple frequencies is NP-hard on general graphs and propose approximation algorithms with worst-case provable performance guarantees for geometric networks. In particular, they design a constant factor approximation algorithm f or unit disk graphs where every node has a uniform transmission range and a $O(\Delta(T)\log n)$ approximation for general disk graphs where nodes have different transmission ranges, where $\Delta(T)$ is the maximum node degree in the routing tree. They also show that a constant factor approximation is still achievable when the routing topology is not known a priori so long as the maximum node degree in the tree is bounded by a constant. To minimize latency, in [32] shows that minimum-length scheduling does not automatically guarantee minimum latency, and a heuristic is proposed to minimize latency by scheduling the incoming links before the outgoing links. In [33], Pan et al. propose algorithms for quick convergecast in ZigBee tree-based WSNs. The objective is to enable quick convergecast operations with minimum latency and complying with the ZigBee standard. Different from other studies, which minimize latency by minimizing the schedule length and assigning slots to the senders, this study considers receiver-based scheduling. This is due to the fixed wake-up/sleep scheduling specified in the ZigBee stack: in each cycle, nodes wake up twice, first to receive packets from their children and second to transmit to their parents in a ZigBee beacon-enabled tree network. The authors first define a minimum latency beacon scheduling problem for quick convergecast in ZigBee networks and prove it to be NP-complete. Then they propose an algorithm that gives optimal performance for tree-based schemes as a heuristic. A centralized tree-based algorithm traverses the nodes on a tree in a bottom-up manner, starting with the leaf nodes. Nodes at the same depth of the tree are sorted according to the interference values (i.e., the number of links that cause interference on the link between the node and its parent) starting with the most interfered node, and scheduling continues sequentially by assigning the first minimum available slot. Finally, a distributed version of the time slot assignment algorithm is proposed.

Among the works to minimize the energy consumption, [34] studies a TDMA scheduling scheme for many-to-one communication is studied. TDMA-based communication provides a common energy-saving advantage by allowing nodes to turn their radio off when not engaged in communication; however, too much state transitions between the active and the sleep modes can waste energy. Accordingly, the desired objectives in this paper are to minimize the total time for data collection as well as to minimize the energy consumed on switching between the active and sleep states. To solve this optimization problem, two population-based stochastic optimization techniques, particle swarm optimization and genetic algorithm, are hybridized. The former guarantees that there is no empty slot during scheduling, and the latter ensures a strong searching ability to find the optimal slot allocation. It is shown by simulations that the hybrid algorithm outperforms the particle swarm optimization algorithm and the coloring methods in terms of the energy efficiency and finding minimal schedule lengths. In [35], ElBatt et al. study the problem of joint scheduling and power control. Although the ideas presented in this paper are not directly targeted for WSNs, the problem of joint power control and TDMA scheduling also arises in WSNs, and the solution presented in the paper has been used for minimizing the data collection time in [30]. The algorithm proposed in [35] is a cross-layer method for joint scheduling and power control to improve the throughput capacity. The goal is to find a TDMA schedule that can support as many transmissions as possible in every time slot. It has two phases: (i) scheduling and (ii) power control. The scheduling phase searches for a valid transmission schedule where no node is to transmit and receive simultaneously or to receive from multiple nodes simultaneously. The power control phase then iteratively searches for an admissible schedule with power levels chosen to satisfy all the interfering constraints in the given valid schedule. In each iteration, the scheduler adjusts the power levels depending on the current RSSI at the receiver and the SINR threshold according to the iterative rule: $\text{Pnew} = \frac{\beta}{\text{SINR}} \cdot \text{Pcurrent}$, which is the well-known power control algorithm by Foschini and Miljanic [36]. If the maximum number of iterations is reached and there are nodes that cannot meet the interfering constraints, the scheduling phase excludes the link with minimum SINR. The power control phase is then repeated until an admissible transmission scenario is found.

ROUTING MECHANISMS: The overlay construction is actually a logical route selection process for each mobile node to send the network state information to the central server. Given the multi-hop and mobility characteristics in our Multinetwork environment, we would like to review some related work on routing mechanism in MANET first. Then we will analyze the similarities and differences between our case and MANET in order to propose our algorithm. There are two main challenges to face when applying routing algorithms on MANET. First, in multi-hop scenario the role of collisions and interferences becomes more complex and depends on many factors such as radio environment, modulation schemes, transmission power, or sensing ranges. To alleviate such problem, in WMNs mesh routers may be equipped with multiple radios to simultaneously transmit/receive over different orthogonal frequency channels. Thus a planned channel allocation should be devised. The second problem is the nodes mobility. The network might become disconnected for a long period or the high mobility might lead to frequently changing communication paths. Therefore, commonly assumed communication design principles such as the permanent availability of a dedicated end to end path have to be reconsidered, leading to new communication paradigms that are significantly more delay tolerant than common approaches (such as digital postal service through store-carry-forward message delivery). Instead of assuming an always-on connection, communication entities rather carry information between intermittent communication opportunities, leading to the opportunistic communication paradigm.

Due to the characteristics of multi-hop communication and the low resource availability in MANET, simply deploying overlay protocol like P2P based protocol as is on top of MANET routing layer might cause poor performance (as stated in [37], [38].) Thus most MANET overlay construction protocols are cross-layer based or integrated with the network layer. We can generally classify these overlay protocols into two categories: unstructured and structured. Unstructured overlays do not impose a rigid relation between the overlay topology and where resources or their indices are stored. The advantages include: simple to implement and to support dynamic environments. The major drawback is scalability. In [39], ORION aims at providing P2P services in a MANET, bring a general purpose distributed lookup service and enhancing file transmission schemes to enable file sharing in MANETs. The basis of ORION is AODV, and it concentrates only on file sharing applications, providing an application layer routing protocol that causes unnecessary overhead. The MPP (Mobile Peer to Peer) protocol [40] is also proposed as a file sharing system in MANETs. In contrast to ORION, MPP adapts the overlay structure to the physical MANET structure via a cross layer communication channel between the MANET network layer and the application layer. In order to reduce the heavy overhead of always broadcasting search requests in the MANET, zone-based protocols, such as ZP2P (Zone-based P2P by [41]) have been proposed. ZP2P is based on the concept of local zones, determined by a fixed hop-count. When a node is interested in a specific resource, it will first check its local cache to see whether any of its zone members can provide the desired object. This process continues until either a predefined TTL expires or the whole network has been searched.

Structured overlay impose a structure on the overlay topology by no longer choosing routing table entries arbitrarily. Instead, routing table entries have to satisfy certain criteria depending on the respective DHTs (Distributed Hash Table). At the core of each DHT lies the ability to route a packet based on a key, towards the node in the network that is currently responsible for the packet's key. This process is referred to as indirect or key-based routing. This structure enables DHTs to introduce an upper bound on the number of overlay hops towards the node currently responsible for the packet's key. This upper bound is commonly O(logn), n being the number of nodes in the overlay network. Some works are building application layer overlay that is transparent to network layer. In [42], the performance of Bamboo is evaluated in a static multi-hop environment common to ad-hoc networks. When deploying Bamboo over MANET following a layered approach, the overlay network forms a virtual network in the application layer while the underlying network is transparently managed by MANET routing protocols such as AODV. Other related publications, such as [43] which deploys Chord over MANET routing protocols, also indicate that simply deploying a standard MANET routing layer does not scale with increasing number of clients, network size, and mobility. The reasons are manifold, such as the characteristics of multi-hop communication, the consistency problem between the two routing layers, and the design assumptions for MANET routing protocols that assume traffic characteristics unlike those of structured overlay protocols. Other works are based on an integrated approach operating both on application layer and network layer. Virtual Ring Routing (VRR) [44] is a networking routing protocol which pushes peer-to-peer concepts to the network layer itself. Caesar et al. argue that VRR brings benefits when implemented over MANETs, as it balances the load of managing hash-table keys across nodes, and avoids flooding of routing messages through the network. VRR organizes the nodes into a virtual ring ordered by their identifiers. Each node maintains a small number of routing paths to its neighbors in the ring. Scalable Source Routing protocol (SSR) [45] brings the same concept of VRR while trying to integrate the P2P overlay into the network layer. But while VRR does not assume any specific MANET routing protocol integration, SRR combines the Dynamic Source Routing protocol (DSR) [46] in the physical network with Chord routing in the virtual ring formed by the address space. It is stated that SSR trades off shortest path for a reduced amount of state information, leading to less maintenance overhead. Therefore, besides the successor, SSR's nodes store the addresses of O(logn) additional nodes at exponentially spaced distances to reduce the average request path length from O(n) to O(logn), where n is the number of nodes in the network. MeshChord, proposed by [47], is a specialization of Chord applied to wireless mesh networks, where the availability of a wireless infrastructure, and the 1-hop broadcast nature of wireless communication are taken into account while performing key lookup. In MeshChord, routers are assumed to be stationary, but they can be switched on/off during network lifetime. If a client in the mesh network wants to find a certain resource, it sends a key lookup message to its reference mesh router (a mesh router within its transmission range). The reference router forwards the resource request in the DHT overlay according to the rules specified by the Chord protocol, until the resource query can be answered. As in Chord, in a n-node system, each MeshChord's node maintains information about only O(logn) other nodes, and resolves lookups via O(logn) messages to other nodes.

Our overlay structure in Multinetwork management system has some different characteristics with the pure MANET. First, every transmission has the central DB as either the source (in case of issuing collection command) or destination (in case of receiving information from mobile nodes). There are no pure Peer-to Peer connections between any two arbitrary mobile nodes. It is more likely a single source multicast when the central DB issues the collection command (queries) to the mobile nodes, and a multiple source-single sink communication when mobile nodes send information back to the central DB. However, since there are heterogeneous and mobile nodes in our scenario, solutions to single source multicast and multiple source-single sink problems are not suitable out-of-box in our case.

WIRELESS SENSOR NETWORK MANAGEMENT: Besides the five main domains in traditional network management system (configuration management, performance management, faulty management, security management and accounting management), there are several points that the wireless sensor network management system should also consider: lightweight operation, robustness, adaptability and responsiveness, minimal storage and scalability. The Multinetwork management system has similar features with wireless sensor network management system in terms of nodes mobility, limited power and computation capability, and sink-sensor/server-nodes communication pattern. MANNA is a policy-based management system that collects dynamic management information, maps this into WSN models, and executes management functions and services based on WSN models. It adopts cluster mechanism as the underlying overlay architecture. Cluster heads are responsible for executing local management functions and they aggregate management data from sensor nodes. Cluster heads forward management data directly to the base station. Furthermore, cluster heads can work cooperatively with other cluster heads to achieve an overall management goal, for example, forming groups of nodes. Considering fault management, MANNA [48] provides two main management services: coverage area maintenance service and failure detection service. The central manager uses the topology map model and the energy model to build a cover-age area model in order to monitor areas of sensing and communication coverage. The central manager can command the agent to execute a failure detection management service, like send GET-RESPONSE message to a node. If there is no response and there is available energy for this node shown in the energy map, a node failure is detected. However there is no fault recovery mechanism in MANNA.

SNMS[49], a Sensor Network Management System, is an interactive system for monitoring the health of sensor networks. SNMS provides two main management functions: query based network data collection and event logging. The event driven logging system allows the user to set event parameters and nodes in the network will report their data if they meet the specified event thresholds. SNMS supports two traffic patterns: collection (for getting health data) and dissemination (for distributing management messages, commands, and queries). SNMS uses a data-gathering tree to collect network information from sensor nodes. This tree construction protocol uses flooding with random staggering of retransmission times for each node. In SNMS, every node in the network only maintains the single best parent based on the strongest received signal strength in order to minimize memory usage. Network states such as a node's current parent and link quality are only updated based on user queries. SNMS employs Drip protocol to disseminate message. They use an identifier to represent a channel, in which each sensor nodes cache and extract data from the latest message received if this node subscribes this channel. The main advantage of SNMS is that it introduces overhead only for human queries and so has minimal impact on memory and network traffic. SNMS further minimizes energy consumption by bundling the results of multiple queries into a single message instead of returning results individually. The main drawbacks of SNMS are that the network management function is limited to passive monitoring only, requiring human managers to submit queries and perform post-mortem analysis of management data. Furthermore, SNMP's centralized processing approach requires continuous polling of network health data from managed nodes to the base station, and this can burden sensor nodes that should minimize transmissions in order to extend network lifetime.

Deb [50] propose a management framework called Sensor Network Management Protocol (sNMP). The sNMP framework has two functions. First it defines sensor models that represent the current state of the network and defines various network management functions. Second, it provides algorithms and tools for retrieving network state through the execution of the network management functions. Models for sensors include network topology (node connectivity), energy map (node battery power), and usage patterns.

sNMP uses TopDisc algorithm, which provides a clustering mechanism that allows a minimal set of nodes to be active, to maintain network connectivity. TopDisc has three management functions: network state retrieval, data dissemination and aggregation, and duty cycle assignment. For example, a node may refuse to be a for-warding node if it has insufficient energy. An advantage of the TopDisc approach is that it provides a framework to perform network management functions based on local information that is highly scalable. However, clustering introduces overheads of cluster-head election and cluster maintenance that are expensive in terms of latency and energy. In addition, sNMP has a distributed parameterized algorithm, STREAM, which is designed to return network topology at a required Resolution, at proportionate costs. It first defines a minimal independent dominating set and use parameter r to indicate the desired network detail scope. The algorithm will return the nodes in the scope of minimal independent dominating set plus r hops. It makes a trade-off between topology details and resources usage.

[51] propose an adaptive policy-based management system for WSNs, called Wireless Sensor Network Management (WinMS).

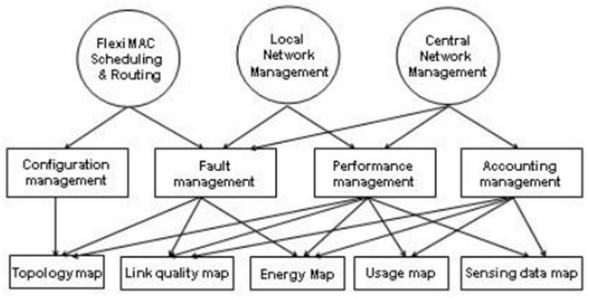


Figure 5

Figure 5 shows the WinMS architechture. FlexiMAC [52] is the underlying MAC and routing protocol that schedules node communication and continuously and efficiently collects and disseminates data, to and from sensor nodes in a data gathering tree. A local network management scheme provides autonomy to individual sensor nodes to perform management functions according to their neighborhood network state, such as topology changes and event detections. The central network management scheme uses the central manager with a global knowledge of the network to execute corrective and preventive management maintenance. The central manager maintains an MIB that stores WSN models that represent network states. The central manager analyses the correlation among WSN models to detect interesting events such as areas of weak network health, possible network partition, noisy areas, and areas of rapid data changes. An advantage of WinMS is that its lightweight TDMA protocol provides energy-efficient management, data transport and local repair. Its systematic resource transfer function allows non-uniform and reactive sensing in different parts of a network, and it provides automatic self-configuration and self-stabilization both locally and globally by allowing the network to adapt to current network conditions without human intervention. A disadvantage of WinMS is that the initial setup cost for building a data gathering tree and node schedule is proportional to network density. However, this one-off cost can be tolerated because nodes maintain the gathered information throughout their lifetime in the network.

Sympathy [53], is a debugging system developed to identify and localize the cause of failures in sensor network applications by collecting network performance metrics with minimal memory overhead, analyzing the metrics to detect events, and identifying spatiotemporal context of the events. When the Sympathy-sink detects an event, it provides a temporal context by retrieving historical metrics associated with the node causing the trigger and also neighbors of the node. The Sympathy sink can probe the root cause of the problem (event) by injecting a request to the network to collect the neighborhood information of that node. Once the Sympathy sink verifies the hypothesis of the root cause, it informs clients interested in that event.

In two-phase self-monitoring system (TP)[54], there are two ways of detect faults: explicit and implicit. Explicit fault detection is performed by nodes themselves analyzing sensor data and triggering an alarm if an event of interest occurs. This scheme offers low energy overhead as the base station expects nothing unless the nodes report event triggers. Implicit fault detection refers to the detection of node communication failures that may be due to energy depletion, intrusion, or environmental factors such as physical damage to nodes. To detect implicit faults, continuous monitoring of sensor nodes is required. TP uses a distributed scheme for monitoring node activity in which nodes perform both implicit (active monitoring) and explicit (passive monitoring) fault detection based on neighborhood information.

3.HIGH LEVEL DESIGN THOUGHTS:

Client slides middleware, Embedded/function cooperative with server or proxy

- Middleware should be installed in the client side
- Add-on function (rather than embedded so that can be configured at run time) is working cooperatively with server or proxy (tier two nodes or more powerful nodes in tier 3)

Cross layer information collection combined with context input

- As input of network selection function
- Send to the central data base

Multiple interface (connector) selection: infra/ad hoc tier 3 convergecast tree and cluster

- Core part of the overlay construction
- Convergecast tree integrated with mobility awareness, to minimize delay and energy.

HIGH LEVEL DESIGN:

The International Organization for Standardization (ISO) network management model defines five functional areas of network management [55]. Those five functional areas provide practical recommendations to increase the overall effectiveness of current management tools and practices. They also provide design guidelines for future implementation of network management tools and technologies.

The ISO network management model's five functional areas are listed below:

- Fault Management—Detect, isolate, notify, and correct faults encountered in the network.

- Configuration Management—Configuration aspects of network devices such as configuration file management, inventory management, and software management.

- Performance Management—Monitor and measure various aspects of performance so that overall performance can be maintained at an acceptable level.

- Security Management—Provide access to network devices and corporate resources to authorized individuals.
- Accounting Management—Usage information of network resources.

By definition, the main goal of fault management is to detect, log, notify users of, and (to the extent possible) automatically fix network problems (i.e., fault recovery) to keep the network running effectively. Because faults can cause downtime or unacceptable network degradation, fault management is perhaps the most widely implemented of the ISO network management elements. Detecting (through monitoring network devices), isolating, notifying, and correcting faults encountered in the network can improve the dependability of the infrastructure. Usually, the following commonly available functions are included in a standard management platform:

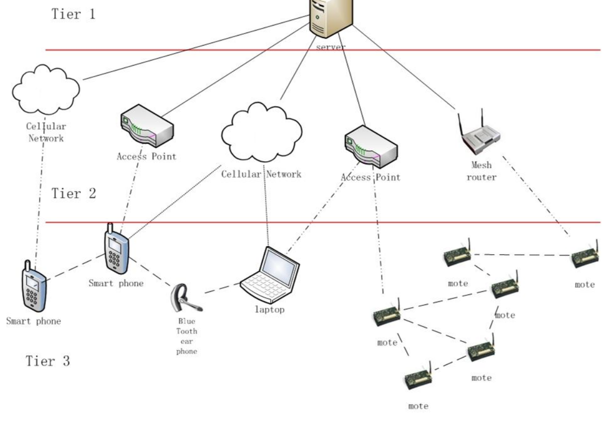
- Network discovery
- Topology mapping of network elements
- Event handler
- Performance data collector and grapher
- Management data browser

Network management platforms such HP OpenView, Computer Associates Unicenter, and SUN Solstice can perform a discovery of network devices. Each network device is represented by a graphical element on the management platform's console. Different colors on the graphical elements represent the current operational status of network devices. Network devices can be configured to send notifications, called SNMP traps, to network management platforms. Upon receiving the notifications, the graphical element representing the network device changes to a different color depending on the severity of the notification received. The goal of configuration management is to monitor network and system configuration information so that the effects on network operation of various versions of hardware and software elements can be tracked and managed. Performance management allows enforcing a service level agreement (SLA) between a service provider and its customers. By definition, a service level agreement (SLA) is a written agreement between a service provider and its customers on the expected performance level of network services. The SLA consists of metrics agreed upon between the provider and its customers. The values set for the metrics must be realistic, meaningful, and measurable for both parties. Various interface statistics can be collected from network devices to measure the performance level. These statistics can be included as metrics in the SLA. Statistics such as input queue drops, output queue drops, and ignored packets are useful for diagnosing performance-related problems. At the device level, performance metrics can include CPU utilization, buffer allocation (big buffer, medium buffer, misses, hit ratio), and memory allocation. The performance of certain network protocols is directly related to buffer availability in network devices. Measuring device-level performance statistics are critical in optimizing the performance of higher-level protocols. The goal of security management is to control access to network resources according to local guidelines so that the network cannot be sabotaged (intentionally or unintentionally). A security management subsystem, for example, can monitor users logging on to a network resource, refusing access to those who enter inappropriate access codes. Since security management is a very broad subject, we are not deeply considering security challenges in our management system, at least initially. Accounting management is the process used to measure network utilization parameters so that individual or group users on the network can be regulated appropriately for the purposes of accounting or chargeback. Similar to performance management, the first step toward appropriate accounting management is to measure the utilization of all important network resources. A usage-based accounting and billing system is an essential part of any service level agreement (SLA). It provides both a practical way of defining obligations under an SLA and clear consequences for behavior outside the terms of the SLA.

All the aforementioned management systems are strongly SNMP-based. SNMP is de-facto standard for traditional wired networks and is still a good reference for Multinetwork management systems. However there are several substantial differences that don't allow us to leverage on SNMP for our scenario. For instance, an SNMP based management system deployed in a wired network can rely on robust end-to-end connection between the management station and the agents running on the managed devices and doesn't need to consider frequent topology changes, i.e. nodes are fixed. Furthermore, it doesn't consider heterogeneous link but, for example, assume that every device is physically connected using the same technology (e.g., Ethernet). In our work, furthermore, we focus mainly on user device management, rather then on pure network devices (e.g., switches, routers), and we consider also mobile devices (e.g., laptops and smartphones). In a Multinetwork scenario, several multimodal devices coexist and can share connectivity resources with neighbors. Thus, communications between nodes can happen through different networks and access technologies (LAN, WLAN, WMN, WPAN, Cellular). Considering those peculiarities, SNMP and other works considered in the previous chapter are not well suitable for our scenario.

The management system needs to create and maintain (dynamically reacting to joining and leaving nodes) an overlay network above the existing heterogeneous links, in order to collect state information about the devices themselves and the links they are communicating through. A logically centralized server is responsible for collecting and storing this information into a database to achieve the management objectives described in the previous section. The system is designed considering the OAA model (Observe, Analyze, Adapt). It receives network state information from the devices, analyzes them, and possibly issues configuration commands, e.g., to recover a previously detected faults. Using this information, it can either coordinate different kinds of networks more efficiently or perform management operations on specific network devices. As stated before, the first step to accomplish in our management system is the creation of the overlay network. In this process, the main idea is to organize different kinds of devices in a hierarchical manner, and to achieve both better scalability and robustness to links/nodes failures due to the node mobility.

The figure above describes how we would like to organize the devices in the overlay network in order to collect state information at the server. The server (that could be also a cluster of machines) is a fixed machine with unlimited power supply and strong computation capabilities. It is located as root of the hierarchy at the Tier 1. At the Tier 2 there are stationary devices (e.g., 802.11 AP, Mesh routers, Base stations) that are connected with a wired connection to the server and offer wireless connectivity to the Tier 3 devices (Infrastructure manner). Tier 3 devices are usually mobile nodes and have limited computation and energy resources. Furthermore, they can be directly connected each other in an Ad-hoc



manner (e.g., 802.11 Ad-Hoc, Bluetooth). This hierarchical overlay allows classifying different kinds of devices (with different properties and resource availability) at different tiers of the hierarchy. The more stable and reliable the devices, the higher they are located in such hierarchical overlay. In this way it is easier to handle node joining and leaving (typical of mobile nodes) because it involves only the lower Tier and has a lower impact on path availability to the server for upper Tier nodes. The network monitoring process is characterized by two main traffic patterns: collection and dissemination. The former is required to obtain state information from the devices; the latter is required to distribute management commands and queries to the devices. Usually, the dissemination phase is followed by the collection one. In order to coordinate those two traffic patterns, our system is deployed both at server and client side. This means that every node that belongs to the Multinetwork environment needs to run our software stack.

As already mentioned, our management system is designed following a client- server architecture (similarly to the SNMP manger-agent model). In the following sections we are going to show more details about the system architecture.

SERVER ARCHITECTURE

The main goal of the software stack running at the server side is to coordinate the overlay network construction and to continuously collect network state information in order to actively monitor the managed environment. Collected state is also persisted into a database; thus is possible to analyze it for multiple purposes, e.g., detect possible faults and unexpected behaviors. The following items describe the main tasks of the server to accomplish the desired management functions.

- Read the configuration file and establish connections with Tier 2 nodes (or dynamic discovery)
- Initiate the overlay construction.
- Invoke the state information (link/node/app state) collection process after disseminate queries to the clients
- Store the state information in DB
- Reason the state and depending on the reasoning results give advices/hints to the nodes about configuration
- Discover faults (explicit/implicit) in the network and depending on the discovery and reasoning results perform fault recovery operations

The following Figure gives the overall architecture of the server side software stack.

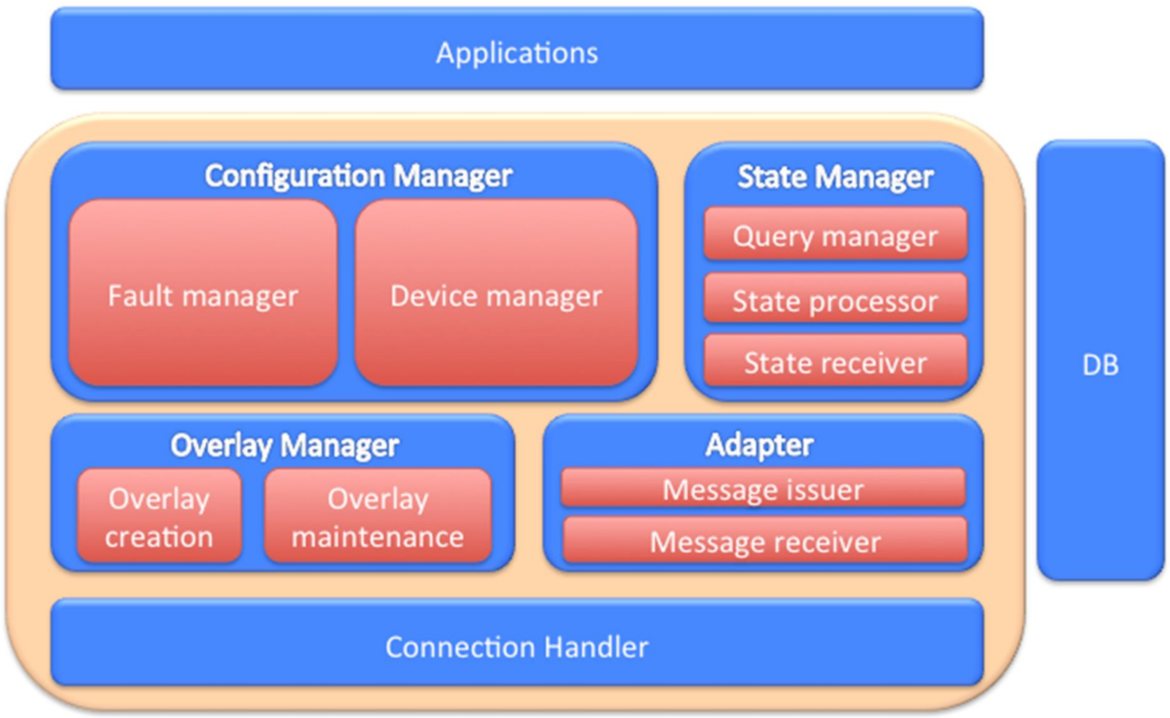


Figure 6

Server architecture is structured in three different layers and is composed by five main components: Connection Handler, Overlay Manager, Adapter, Configuration Manger, and State Manager.

Connection Handler is the lowest layer component in our architecture. It creates an abstraction layer that hides TCP/IP mechanism details from the upper layers and handles incoming connections from the lower tiers nodes. At the same time, it allows the server to communicate with client nodes. In a nutshell, this component provides some communication APIs to the upper layer components, simplifying the implementation of the application protocols for the state collection and query dissemination. It basically coincides with RAMP [56] Core API.

Overlay Manager is responsible for creating (Overlay creation) and maintaining (Overlay maintaining) the overlay network that enables devices and server to communicate each other. The main goal of this component is to maintain a global picture of the Multinetwork topology. It stores information about the Tier 2 nodes so that it is easy to reach them when needed. This proactive approach could lead to believe that there is too much overhead in maintaining up-to-date this kind of information in an evolving environment. However, let us stress that, by hypothesis, nodes in Tier 2 are fixed. Thus it is unlikely that is needed to continuously refresh topology information for those nodes. We will describe the overlay algorithm in finer details in the next sections.

Adapter enables connection multiplexing and demultiplexing, allowing upper layer components to send and receive packets possibly sharing the same physical connection (connection pooling). Message issuer receives messages from the upper layer components, adapts them to the dissemination protocol, and is responsible to route them to the destination component at the client nodes. To this purpose, it interacts with the Overlay manager to discover the next hop in the overlay. On the other hand, Message receiver receives incoming packets from the Connection handler and dispatches them to the destination component depending on the message type.

State Manager is one of the core components of the entire system. State receiver collects state information from the Tier 2 and Tier 3 devices, and stores them in the database. State processor is responsible for analyzing the state collected and, depending on the results of state analysis, can delegate the Fault manager (see below Configuration Manager) to handle possible faults (implicit fault detection). Finally, Query manager is responsible to manage queries issued by the applications either by retrieving info from the database (previously stored) or by probing client nodes, either in case the asked information hasn't already been persisted or it is incomplete or out-of-date.

Configuration Manager is composed by Fault manager and Device manager. The former is in charge to detect explicit/implicit faults and tries to recover them possibly using the latter. By explicit faults we mean alarms and events triggered by the client devices themselves when a local condition is satisfied, e.g., when the energy level falls below a given threshold. By implicit faults, instead, we mean unwanted network conditions, inferred by the state collection analysis made by State processor (see above State Manager), which usually have a more global scope, e.g., when a network is congested or a given AP is serving too many devices. Device manager is used to issue configuration commands to the client devices. Commands (or "hints") can be used either after a fault detection, thus for trying to recover it, or to suggest a new configuration to the target devices (e.g., turning on/off a network interface for power saving, switch among operational modes, suggestion about overlay construction or path selection).

On top of our Multinetwork management system, existing (or yet to be developed) applications can leverage on the services provided to fulfill their requirements.

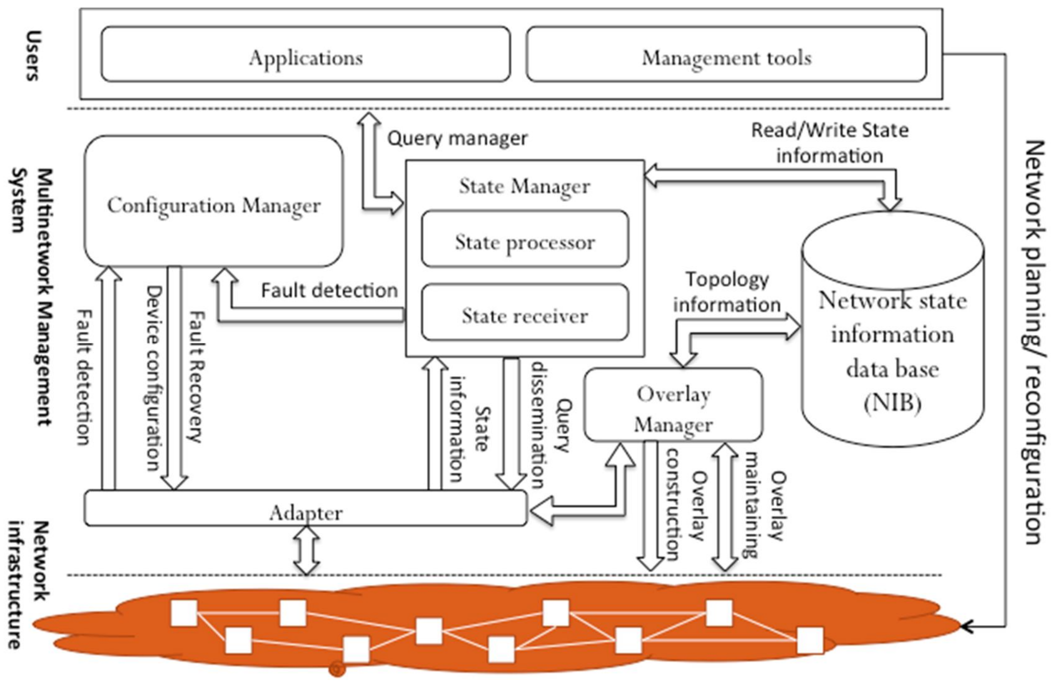


Figure 7

The Figure above shows more details about the interactions between the server components described in the previous paragraphs.

CLIENT ARCHITECTURE

As we stated before, part of the management system is deployed also in the managed devices in Tier 2 and Tier 3. In this section we are going to describe the fundamental components that realize the client architecture. The wide goals of the client software stack include collaborating with the server in order to send state information about the managed device. At the same time, every client is ready to receive configuration hints in order to (try to) recover from faults or to improve the overall network performance. Since we are considering heterogeneous kind of device (that also affects the hierarchical overlay), client architecture is adaptable and flexible, depending on the kind of device it is deployed on. The main difference between the stack deployed on Tier 2 and Tier 3 devices, for instance, is about in-network state aggregation. Considering by assumption more resource availability (computation and energy) on Tier 2 nodes, they can aggregate the state information from Tier 3 nodes (e.g., for reducing bandwidth overhead). State aggregation clearly leads to an approximate global state at the server, but it is necessary in order to achieve a reasonable tradeoff between fine-grained state collection (with optimal analysis results) and resources utilization. Furthermore, in order to monitor Wireless Sensor Networks (WSNs), some client devices will be used as proxy to flow (aggregated) state information to the server. Main functions that a client is responsible of are summarized as follows:

- Discover and establish connections with neighbors in a decentralized way
- Choose parent and accepts children (overlay construction)
- Send local network state (link/node/app) towards the server
- Configure several event thresholds to perform explicit fault discovery
- Cooperate with the server to configure local parameters (overlay, energy, etc.) and fault recovery.

The following Figure gives the overall architecture of the client side software stack.

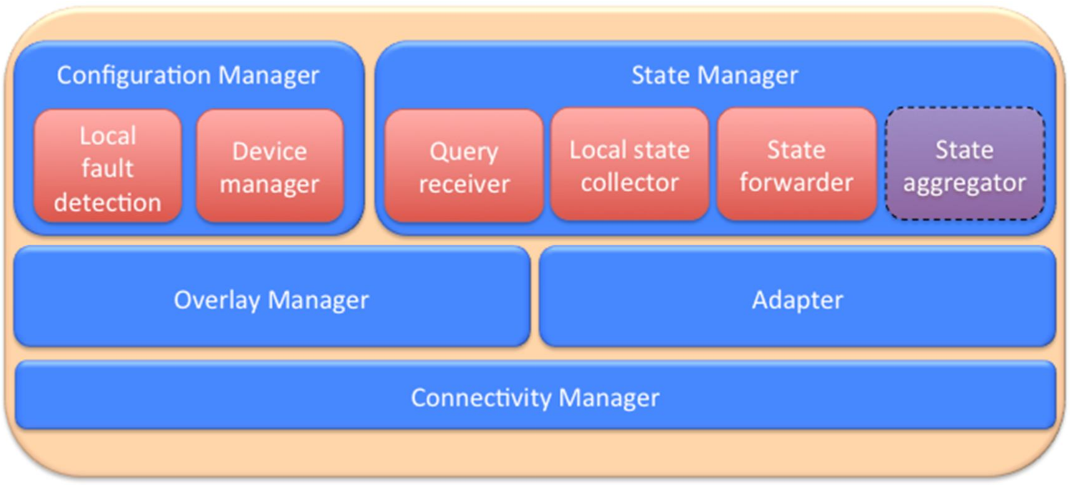


Figure 8

Client architecture is structured in three different layers and is composed by five main components: Connectivity Manager, Overlay Manager, Adapter, Configuration Manger, and State Manager. It is possible to notice that client and server architecture are quite symmetric.

Connectivity Manager represents the lowest layer component in the client side architecture and is responsible to interact with the TCP/IP stack, creating an abstraction layer to the upper components. Each device is supposed to have more than one network interface and this component hides the heterogeneity providing a common API for communicating with neighbor nodes, regardless the specific kind of physical link used. In order to make a node able to communicate over heterogeneous links, it is first necessary to establish a physical connection to an available wireless connector (e.g., 802.11 AP/Ad-hoc, Bluetooth). To accomplish this step, Connectivity manager is able to dynamically discover new network opportunities (either Infrastructure or Ad-hoc) and tries to connect to them according to a given policy (e.g., choosing the connector with best signal quality). Most of this component is based on RAMP [3] Core API.

Overlay Manager is responsible for running the overlay construction algorithm in order to choose the parent and children nodes in the overlay tree. This component is also responsible to react to node joining/leaving by repairing possible interrupted paths to the server. The overlay construction algorithm is strictly related to the path selection mechanism. To send state information to server, for instance, every client just needs to forward messages to the parent node in the hierarchy.

Adapter is basically the same as on the server side. It is used for multiplexing and demultiplexing messages sharing the same physical connection. It receives messages from the upper layer components and sends them through the Connectivity Manager. On the other hand, when it receives a message from another node it dispatches it to the right component depending on the message type.

Configuration Manager is responsible for detecting local faults (Local fault detection) and configuring the device as suggested by the server (Device manager). Local fault detection basically identifies unwanted behaviors and triggers explicit alarms to the server (e.g., battery level below a given threshold, link failure, etc.). Device manager, instead, receives configuration commands/hints from the server and tries to apply them to the current configuration.

State Manager in the client side is composed by Query receiver, Local state collector, State forwarder, and State information aggregator. Query receiver handles incoming queries from the server and coherently sets up state forwarding policies (update frequency, data accuracy, data attributes) in order to achieve a tradeoff between state accuracy and bandwidth consumption (i.e., fine-grained or coarse-grained state collection). Local state collector works locally to gather local node information (e.g., battery level, links state) to be forwarded to the server according to the queries received. State forwarder sends the information previously obtained from the Local state collector to the server according to the related policies. It is also responsible to propagate state information received from lower tier nodes up in the hierarchical overlay (store-carry-forward model). Forwarding policies are critical to reach an effective tradeoff between proactive and reactive (on-demand) state forwarding. Forwarding state information in a proactive manner means that a node sends some local information on its own initiative, without being asked explicitly from the server. In this case we may have a lower latency to answer to a specific set of queries (because information is already persisted into the server database) but we also have a bigger overhead because those information may not be always meaningful. On the other hand, using a reactive (on-demand) state forwarding means that a node sends local information only when explicitly asked from the server (e.g., after receiving a query). In this case, however, we minimize the overhead at the expense of the latency. Finally, State information aggregator is the most relevant difference between the stack deployed on Tier 2 and Tier 3 nodes. On Tier 2 nodes, indeed, state information received from lower tier nodes may be aggregated (with SUM, MAX, MIN, AVG functions) before being forwarded to the server.

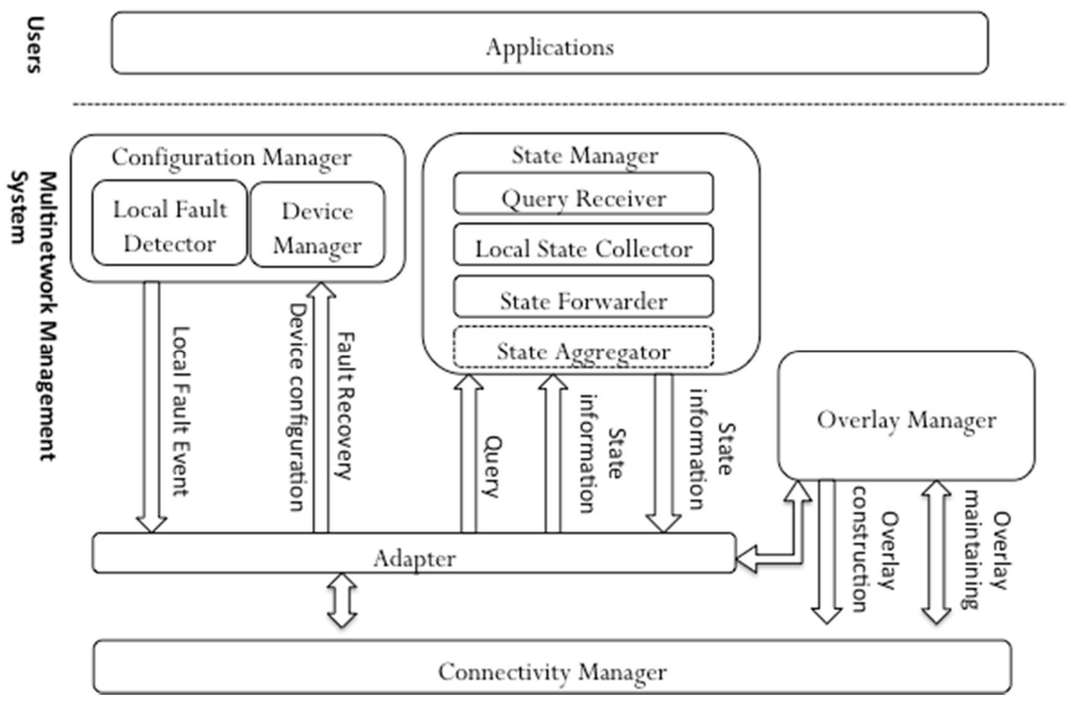


Figure 9

The Figure above shows more details about the interactions between the client components described in the previous paragraphs.

4. OVERLAY CONSTRUCTION PROBLEM FORMULATION AND OBJECTIVES:

Given a graph G(V, E) and a source node s$\in V$ each node (except s) needs to find a path towards s to report its state information to the server. s needs to find a path to reach every other node in case the server wants to issue queries, commands, and hints to mobile nodes. The objectives of the overlay construction are: best path selection, low messaging overhead, low memory occupation, quick convergence time and scalability.
Best path selection means that the overlay route selected by our algorithm between the server and the mobile node is best in terms of delay (initially we assume that delay is proportional to the number of hops between two nodes). Low consumption of messages means to construct and maintain this overlay network does not cause huge messages exchange that may affect other applications (because the bandwidth of wireless channels are not high). Low memory occupation means that mobile nodes do not need to store much information to maintain such an overlay (mobile nodes have limited memory). Quick convergence time means that the overlay should response to the change of the underlying network topology quickly (mobile nodes moving around causes great dynamicity).
Scalability means that overhead introduced by the overlay algorithm does not increase

proportionally to the number of nodes (it should be a polynomial function of nodes number or less).
In this section, we only proposed those objectives from a high level perspective. In algorithm part and evaluation part we will demonstrate how we achieve those goals from both theoretical and experimental perspective.

ALGORITHMS:

DATA STRUCTURES:

variables/classes	Type	Description
Neighbor	NodeID:IsParentCand:IsParent:IsChild	One hop Neighbor, with relationship.
NeighborList	List of Neibhors	List of all the one hop neighbors
Child	NodeID	Stores the node id of one of its child. Each node has several children
Descendants	NodeID:NodeIDSet	Stores one of its children and all the descendants of this child
DescendantList	DescendantList	Stores all the descendants
Generation	Int	Generation of this node

INITIALIZATION:

PHASE1:

The source s will first set its generation 0 (infinite by default), and then send a Parent Claim broadcast message to all of its one hop neighbors to advertise itself as possible parent. When a node n receives this message, it will send a Parent Request messages to s. When s receives the Parent Request messages, it will respond with a Parent Confirmation message to n. Once confirmed, n will send Parent Claim broadcast messages to all of its neighbors, in turn. Note that some nodes may receive more than one Parent Claim Messages; in this case they first mark those nodes as parent candidates and then choose the best parent (in terms of generation so far, may be extended later) among them and send the Parent Request message.

PHASE 2:

Once a node has more than one child, it will send a NEW Message to its parent by saying which nodes are its children. This parent will then add a record and forward to the upper level parent, until reach the source. Note that a node only knows its children's id and all the descendants of each child. It does not need to know the exact topology of its descendants below its children. Hence we do not need to consume much memory and bandwidth to maintain a complete topology map in each node.

Figure 1 gives an example of the algorithm initialization:

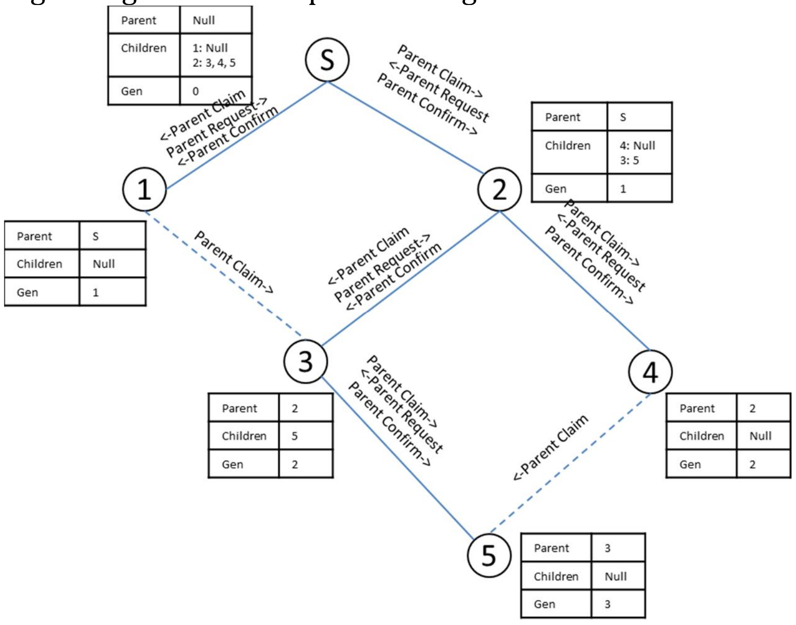


Figure 10

MAINTENANCE:

PARENT LOST:

Every node is scheduled to periodically broadcast heartbeat message to children. If a node does not receive a heartbeat message from its parent after a timeout, it will infer that this parent is unreachable. So it will reselect a parent from its non-children neighbors and compare their generations. If the new parent candidate has a smaller (or equal) generation than this node, it will select this parent as the new parent and set its new generation as the parent's generation plus one. It will also send to all the children the updated generation information, if necessary. If there is no available parent candidate with a smaller or equal generation, this node will notify all its children that the ancestor is failed. Every descendant will reselect its parent. By doing so, we can avoid the so called routing loop problem, which happens when a node chooses one of its descendant as its new parent.

CHILDREN LOST:

Every child will give a response when it receives the heartbeat message from its parent. In the meanwhile, if a new child joins the overlay the parent will add it into the children list immediately. So the parent will refresh the exact children list periodically, and it should report that information upwards, so that when the source node want to send something to a specific node it will know where to send it. Here we use a mechanism that allows us to limit the upward messages propagation. On one hand we can make sure that if an ancestor want to send something to its descendant it will know which children should be the next hop; on the other hand, the ancestor does not need to know an exact map of all its descendants due to the huge message overhead and delay. Our basic idea is that once a new node is leaving the old parent and joining the new one, the topology update message should be only propagated until the most recent ancestor of the new and old parents. The problem is equivalent to how to find the most recent ancestor in a distributed manner. Remember that each node has a children list and it knows the descendants set under each child. If a node has a new joining or leaving child, it will report this to its parent and the parent will edit the descendants set of this node correspondingly. We argue that the joining event can be detected immediately while the leaving event may be found after a timeout, so we mainly use joining messages to figure out which node is the most recent ancestor. When a node finds a new child joining, it will report this to its parent. When a parent receives this information, it will check whether this newly joined node already existed in other child descendant set or not. If yes, this parent is the most recent ancestor and it stops reporting this information further. If not, it will forward this information to its own parent and so on. Figure 2 shows the procedure:

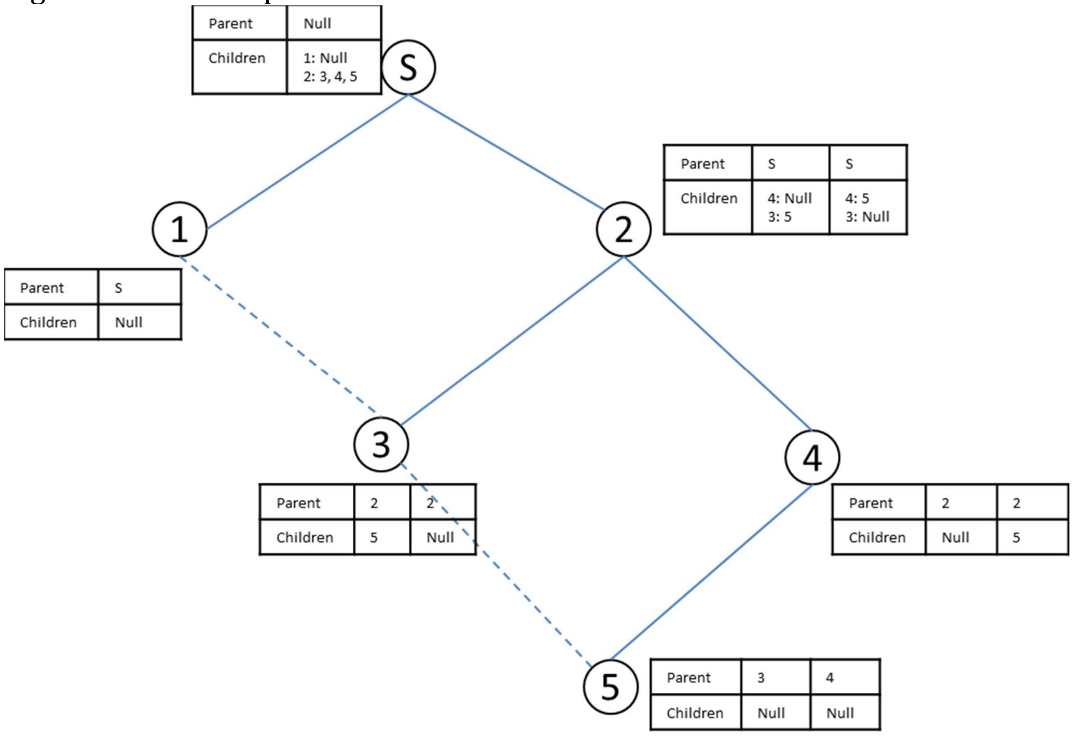


Figure 11

When Node 5 leaves from Node 3 and joins in Node 4, it will update its parent from 3 to 4. Node 3 and 4 will also update their children list from the old state to the new state, shown in the second column and third column respectively. Finally, Node 4 will report this change to Node 2, which finds out that the newly joined node under Node 4 was its descendant under another branch (under Node 3 in this case). In this case it will not report anything to its parent since it knows that it is the most recent common ancestor of Node 3 and Node 4.

STATE TRANSITION GRAPH:

Initially, every node is not part of the overlay network. Once it receives the Parent Claim broadcast messages from several parent candidates it will choose the best one and send Parent Request message to it. Once a parent candidate receives a Parent Request message, it will send the Parent Confirm message to this child and add it into its children list.

When a node receives the Parent Confirm message, it knows it is connected to a path that can lead to the source node. So it will broadcast Parent Claim broadcast messages to all its neighbors. Once it receives the Parent Request message, it will issue a Parent Confirm message to this neighbor and add it into its children list. In addition, after adding a child into the children list, the parent should report this child's id to its parent (this child's grandparent) so that this grandparent can build up a descendant set for each child, which is the basis of children dynamicity management.

When a node finds out its parent is no longer reachable, it will reselect a new parent by checking the Parent Claim messages received from other neighbors and update its generation. If this generation is the same as the previous one, it will do nothing. If the generation is smaller than the previous one, if will simply notify all the children that the generation has been changed. If the generation is bigger than the previous one, it will send the fail messages to all of its children, which means all its descendants should reselect parents.

When a node finds out there is a new child joining, it will send a NEW message to its parent to announce that it has a new descendant. Upon receiving this NEW message, the parent will check whether this newly joined node was already one of its descendant (but located in other branch). If yes, the parent just updates its children list, otherwise it also forwards this NEW message to its parent and so on.

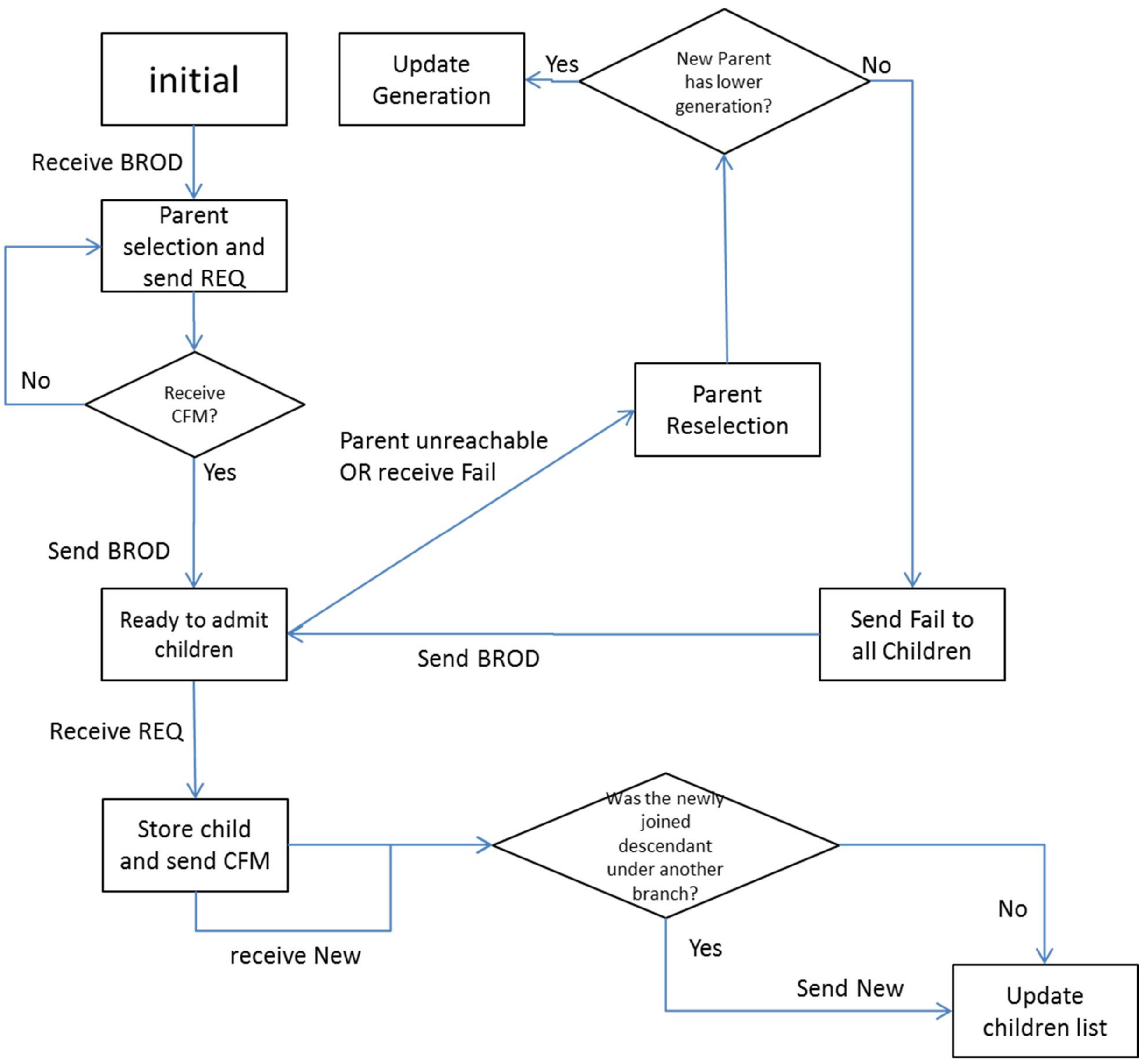


Figure 12

PSEUDO CODE:

```
• Node(i){

NodeInit(i);
while True do
  if pktQueue=/= empty;//cap packet + inferface
    for each pkt in pktQueue do
       PktProcess(i,pkt);
  if(firsttime){SelectParent(i);
  Checkparent(i);
  ParentClaim(i);
```

```
}
    • NodeInit(i){

    Ni=Null;//neighbor list
    Ci=Null;//children list
    i.Generation=INF;
    i.Parent=Null;
    i.NumOfChild=0;
    if(i==s){//node i is the source
       i.Generation=0;
       broadcast(BROD,
              ID=i,
              IP=i.IP,
              Gen=i.Generation,
              NumOfChild=i.NumOfChild);
    }
}
    • PktProcess(i,pkt){

    if(pkt.type==BROD)
          Ni.AddEntry(pkt.id, pkt.ip, pkt.Gen, pkt.NumOfChild, pkt.SINR, pkt.Interface);
    if(pkt.type==REQ)
          i.NumOfChild++;
          Ci=Ci/\pkt.id;
         creat ci.pkt.id.DSet.;
        send(CFM, pkt.ip);
         descendant=pkt.ip;
         send(NEW, pkt.ip, i.parent);
    if(pkt.type==Fail)
        i.generation=INF;
         remove i.parent from Ni;
          SelectParent(i);
    if(pkt.type==UpGen)
          i.generation=pkt.value+1;
          for each child in ci
                  UpdateGeneration(child);
    if(pkt.type==NEW)
          add descendant into pkt.ip.Dset
         if(cannot find descendant in other Dset)
            send(new, descendant, i.parent)
}
    • SelectParent(i){

              i.Parent=best nodes in Ni;//based on some metric
               Send(REQ, i.parent.ip);
               if(receive CFM in timeout){
                if(i.generation>=i.parent.generation){//note 2
           i.generation=i.parent.generation+1;
            if (Ci!=NULL)  UpdateGeneration(i); //note 3
             else ParentClaim(i);
          }else
           send(Fail, Ci.ip)//note 1
        }
         else
             remove i.parent from Ni
             SelectParent(i);
}
   •  ParentClaim(i){

  broadcast(BROD,
              ID=i,
              IP=i.IP,
              Gen=i.Generation,
              NumOfChild=i.NumOfChild,
              SINR=i.SINR)
}
   •  Checkparent(i){

      if i.parent is unreachable
                 remove i.parent from Ni;
                  SelectParent(i);
}
   •  UpdateGeneration(i){

      send(UpGen, ci.ip, i.generation);
}
```

References:

1. Tao, J., G. Noubir, and S. Bo. *WiZi-Cloud: Application-transparent dual ZigBee-WiFi radios for low power internet access*. in *INFOCOM, 2011 Proceedings IEEE*. 2011.
2. Ngoc Minh, D., et al. *Massive live video distribution using hybrid cellular and ad hoc networks*. in *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2011 IEEE International Symposium on a*. 2011.
3. Frank, H., et al., *Delivery of Broadcast Services in 3G Networks.* Broadcasting, IEEE Transactions on, 2007. **53**(1): p. 188-199.

4. Agarwal, Y., et al., *Wireless wakeups revisited: energy management for voip over wi-fi smartphones*, in *Proceedings of the 5th international conference on Mobile systems, applications and services*. 2007, ACM: San Juan, Puerto Rico. p. 179-191.
5. Fadlullah, Z.M., et al., *Toward intelligent machine-to-machine communications in smart grid.* Communications Magazine, IEEE, 2011. **49**(4): p. 60-65.
6. Bellavista, P., A. Corradi, and C. Giannelli, *A Unifying Perspective on Context-Aware Evaluation and Management of Heterogeneous Wireless Connectivity.* Communications Surveys & Tutorials, IEEE, 2011. **13**(3): p. 337-357.
7. Cheng Wei, L., et al., *A framework of handoffs in wireless overlay networks based on mobile IPv6.* Selected Areas in Communications, IEEE Journal on, 2005. **23**(11): p. 2118-2128.
8. Hung-yu, W. and R.D. Gitlin, *Two-hop-relay architecture for next-generation WWAN/WLAN integration.* Wireless Communications, IEEE, 2004. **11**(2): p. 24-30.
9. Luo, H., et al., *UCAN: a unified cellular and ad-hoc network architecture*, in *Proceedings of the 9th annual international conference on Mobile computing and networking*. 2003, ACM: San Diego, CA, USA. p. 353-367.
10. Kang, S.-S. and M. Mutka, *A mobile peer-to-peer approach for multimedia content sharing using 3G/WLAN dual mode channels.* Wireless Communications and Mobile Computing, 2005. **5**(6): p. 633-645.
11. Akyildiz, I.F., S. Mohanty, and X. Jiang, *A ubiquitous mobile communication architecture for next-generation heterogeneous wireless systems.* Communications Magazine, IEEE, 2005. **43**(6): p. S29-S36.
12. Jun-Zhao, S., et al., *Adaptive connectivity management middleware for heterogeneous wireless networks.* Wireless Communications, IEEE, 2005. **12**(6): p. 18-25.
13. Vidales, P., et al., *Autonomic system for mobility support in 4G networks.* Selected Areas in Communications, IEEE Journal on, 2005. **23**(12): p. 2288-2304.
14. Karetsos, G.T., et al., *A hierarchical radio resource management framework for integrating WLANs in cellular networking environments.* Wireless Communications, IEEE, 2005. **12**(6): p. 11-17.
15. Stemm, M. and R. Katz, *Vertical handoffs in wireless overlay networks.* Mobile Networks and Applications, 1998. **3**(4): p. 335-350.
16. Minji, N., et al. *WISE: energy-efficient interface selection on vertical handoff between 3G networks and WLANs*. in *Personal, Indoor and Mobile Radio Communications, 2004. PIMRC 2004. 15th IEEE International Symposium on*. 2004.
17. Mohanty, S. and I.F. Akyildiz, *A Cross-Layer (Layer 2 + 3) Handoff Management Protocol for Next-Generation Wireless Systems.* Mobile Computing, IEEE Transactions on, 2006. **5**(10): p. 1347-1360.
18. Qian, Z., et al., *Efficient mobility management for vertical handoff between WWAN and WLAN.* Communications Magazine, IEEE, 2003. **41**(11): p. 102-108.
19. Stavroulaki, V., et al., *Equipment management issues in B3G, end-to-end reconfigurable systems.* Wireless Communications, IEEE, 2006. **13**(3): p. 24-32.
20. Hongyang, B., H. Chen, and J. Lingge. *Performance analysis of vertical handover in a UMTS-WLAN integrated network*. in *Personal, Indoor and Mobile Radio Communications, 2003. PIMRC 2003. 14th IEEE Proceedings on*. 2003.

21. Veeravalli, V.V. and O.E. Kelly, *A locally optimal handoff algorithm for cellular communications.* Vehicular Technology, IEEE Transactions on, 1997. **46**(3): p. 603-609.
22. Maturino-Lozoya, H., D. Munoz-Rodriguez, and H. Tawfik. *Pattern recognition techniques in handoff and service area determination*. in *Vehicular Technology Conference, 1994 IEEE 44th*. 1994.
23. Nishith, D.T.Z.C.J.H.R.Z.C.H.F.V., *Generic adaptive handoff algorithms using fuzzy logic and neural networks*. 1997, Virginia Polytechnic Institute and State University. p. 262.
24. Hou, J. and D.C. O'Brien, *Vertical handover-decision-making algorithm using fuzzy logic for the integrated Radio-and-OW system.* Wireless Communications, IEEE Transactions on, 2006. **5**(1): p. 176-185.
25. Chen, Y. and Y. Yang, *A new 4G architecture providing multimode terminals always best connected services.* Wireless Communications, IEEE, 2007. **14**(2): p. 36-41.
26. Nasser, N., A. Hasswa, and H. Hassanein, *Handoffs in fourth generation heterogeneous networks.* Communications Magazine, IEEE, 2006. **44**(10): p. 96-103.
27. Xing, B. and V. Nalini. *Multi-constraint dynamic access selection in always best connected networks*. in *Mobile and Ubiquitous Systems: Networking and Services, 2005. MobiQuitous 2005. The Second Annual International Conference on*. 2005.
28. Qingyang, S. and A. Jamalipour, *Network selection in an integrated wireless LAN and UMTS environment using mathematical modeling and computing techniques.* Wireless Communications, IEEE, 2005. **12**(3): p. 42-48.
29. Wei, S., Z. Weihua, and C. Yu, *Load balancing for cellular/WLAN integrated networks.* Network, IEEE, 2007. **21**(1): p. 27-33.
30. Incel, O.D. and B. Krishnamachari. *Enhancing the Data Collection Rate of Tree-Based Aggregation in Wireless Sensor Networks*. in *Sensor, Mesh and Ad Hoc Communications and Networks, 2008. SECON '08. 5th Annual IEEE Communications Society Conference on*. 2008.
31. Ghosh, A., et al. *Multi-channel scheduling algorithms for fast aggregated convergecast in sensor networks*. in *Mobile Adhoc and Sensor Systems, 2009. MASS '09. IEEE 6th International Conference on*. 2009.
32. Djukic, P. and S. Valaee. *Link Scheduling for Minimum Delay in Spatial Re-Use TDMA*. in *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*. 2007.
33. Pan, M.-S. and Y.-C. Tseng, *Quick convergecast in ZigBee beacon-enabled tree-based wireless sensor networks.* Comput. Commun., 2008. **31**(5): p. 999-1011.
34. Mao, J., Z. Wu, and X. Wu, *A TDMA scheduling scheme for many-to-one communications in wireless sensor networks.* Comput. Commun., 2007. **30**(4): p. 863-872.
35. ElBatt, T. and A. Ephremides, *Joint Scheduling and Power Control for Wireless Ad Hoc Networks.* IEEE Transactions on Wireless Communications, 2004. **3**(1): p. 74-85.
36. Foschini, G.J. and Z. Miljanic, *A simple distributed autonomous power control algorithm and its convergence.* Vehicular Technology, IEEE Transactions on, 1993. **42**(4): p. 641-646.
37. Schollmeier, R., I. Gruber, and M. Finkenzeller, *Routing in Mobile Ad-hoc and Peer-to-Peer Networks A Comparison*, in *Revised Papers from the NETWORKING 2002*

Workshops on Web Engineering and Peer-to-Peer Computing. 2002, Springer-Verlag. p. 172-186.

38. Rhea, S., et al., *Handling churn in a DHT*, in *Proceedings of the annual conference on USENIX Annual Technical Conference*. 2004, USENIX Association: Boston, MA. p. 10-10.
39. Klemm, A., C. Lindemann, and O.P. Waldhorst. *A special-purpose peer-to-peer file sharing system for mobile ad hoc networks*. in *Vehicular Technology Conference, 2003. VTC 2003-Fall. 2003 IEEE 58th*. 2003.
40. Gruber, I., R. Schollmeier, and W. Kellerer. *Performance evaluation of the mobile peer-to-peer service*. in *Cluster Computing and the Grid, 2004. CCGrid 2004. IEEE International Symposium on*. 2004.
41. W. Kellerer and R. Schollmeier. *Proactive search routing for mobile peer-to-peer networks:*

Zone-based p2p. in *ASWN*. 2005.

42. Castro, M.C., et al. *Performance Evaluation of Structured P2P over Wireless Multi-hop Networks*. in *Sensor Technologies and Applications, 2008. SENSORCOMM '08. Second International Conference on*. 2008.
43. Cramer, C. and T. Fuhrmann, *Performance evaluation of chord in mobile ad hoc networks*, in *Proceedings of the 1st international workshop on Decentralized resource sharing in mobile computing and networking*. 2006, ACM: Los Angeles, California. p. 48-53.
44. Caesar, M., et al. *Virtual Ring Routing: Network Routing Inspired by DHTs*. in *ACM. Sigcomm*. 2006. Pisa, Italy.
45. T. Fuhrmann, et al., *Pushing chord into the underlay: Scalable*

routing for hybrid manets, in *Technical report, Universitt Karlsruhe (TH)*. 2006.

46. Johnson, D. and D. Maltz, *Dynamic Source Routing in Ad Hoc Wireless Networks*, in *Mobile Computing*, Imielinski and Korth, Editors. 1996, Kluwer Academic Publishers.
47. Burresi, S., et al. *MeshChord: A Location-Aware, Cross-Layer Specialization of Chord for Wireless Mesh Networks (concise contribution)*. in *Pervasive Computing and Communications, 2008. PerCom 2008. Sixth Annual IEEE International Conference on*. 2008.
48. Ruiz, L.B., J.M. Nogueira, and A.A.F. Loureiro, *MANNA: a management architecture for wireless sensor networks.* Communications Magazine, IEEE, 2003. **41**(2): p. 116-125.
49. Tolle, G. and D. Culler. *Design of an application-cooperative management system for wireless sensor networks*. in *Wireless Sensor Networks, 2005. Proceeedings of the Second European Workshop on*. 2005.
50. Nath, B.D.a.B. *http://www.research.rutgers.edu/~bdeb/sensor_networks.html*. 2005.
51. Lee, W.L., A. Datta, and R. Cardell-oliver, *WinMS: Wireless Sensor Network-Management System, An Adaptive Policy-Based Management for Wireless Sensor Networks*. 2006, The University of Western Australia.
52. Lee, W.L., A. Datta, and R. Cardell-Oliver. *FlexiMAC: A flexible TDMA-based MAC protocol for fault-tolerant and energy-efficient wireless sensor networks*. in *Networks, 2006. ICON '06. 14th IEEE International Conference on*. 2006.
53. Ramanathan, N., E. Kohler, and D. Estrin, *Towards a debugging system for sensor networks.* International Journal of Network Management, 2005. **15**(4): p. 223-234.

54. Hsin, C. and M. Liu, *Self-monitoring of wireless sensor networks.* Computer Communications, 2006. **29**(4): p. 462-476.
55. Cisco (2008) *Network Management System: Best Practices White Paper*.
56. Bellavista, P., A. Corradi, and C. Giannelli, *The real Ad-hoc Multi-hop Peer-to-peer (RAMP) middleware: An easy-to-use support for spontaneous networking*, in *Proceedings of the The IEEE symposium on Computers and Communications*. 2010, IEEE Computer Society. p. 463-470.