Probabilistic Decentralized Network Management

Marcus Brunner, Dominique Dudkowski, Chiara Mingardi, and Giorgio Nunzi
NEC Laboratories Europe, Network Research Division, Heidelberg, Germany
Email: {brunner|dudkowski|mingardi|nunzi}@nw.neclab.eu

***Abstract*— This work proposes a probabilistic management paradigm for solving some major challenges of decentralized network management. Specifically, we show how to cope with 1) the overhead of redundant information gathering and processing, 2) the decentralized management in dynamic and unpredictable environments, and 3) the considerable effort required for decentralized coordination of management functions.**

To this end, we describe a framework for probabilistic decentralized management in the context of *in-network management*. We demonstrate how this framework can be applied to a network of information, a novel clean-slate approach towards an information-centric future Internet. We show by means of a simulation study in the area of performance and fault management that we can significantly reduce the effort and resources dedicated to management, while we are able to achieve a sound level of accuracy of the overall network view.

***Index Terms*—Network Management, Probabilistic Management, Decentralized Management, Self-managing systems, Future Internet Management.**

I. Introduction

THIS paper addresses problems in network management systems that are decentralized and need to operate in more or less dynamic networking environments. We assume that future internetworking technology will be managed in a more automated and decentralized way than today. This basically means that management functionality will get nearer to the network functions running in the network, no matter on what layer they are running, or whether the layering principle will persist in future network architectures. In the most extreme case, the management functionality could get conflated with the network functionality, allowing for inherently self-managing networks. But also any degree from the traditional centralized to the fully decentralized, inherently self-managing system is possible, and any mixture thereof. Note that some research explores the opposite direction. McKeown et al. [1], for instance, try to remove as much control and management functionality as possible from the network.

Decentralized management functions typically accumulate management information from the network, then store and process the information for analyzing the history, deriving conclusions, and taking actions eventually. Management functions also coordinate with the same or other types of functions on the same or other nodes. Storage, computation, and communication require resources of the network and the nodes. When all nodes execute those functions, potentially large quantities of CPU time, memory capacity and communication bandwidth will be wasted.

In dynamic and unpredictable network and system environments, management in a traditional way is difficult. The fact that nodes might join and leave the network and might not be reachable from a central station, and that their behavior is brittle, makes the network fairly difficult to manage. Basically, dynamic behavior does not allow for an exact real-time view of the system and it prevents coordinative functionality when the dynamics are too high. Many coordinated decentralized algorithms, for instance, routing protocols or peer-to-peer systems, fail in such dynamic environments. Finally, some cooperating or coordinating decentralized control and management functions tend to converge to a synchronous behavior causing problems in resource usage peaks. We argue that management systems that forego coordination might perform better in such scenarios.

To this end, we propose to use probabilistic decentralized network management in the context of an in-network management framework to tackle the challenges described above. We assume throughout this paper a high degree of decentralization of management functions. Additionally, we assume the management system being decomposed into a set of different management functions running on each node simultaneously. We propose a randomization process being part of a meta-management on each node, which randomly turns on or off certain management functions on the node. This approach is a prerequisite for resource-efficient decentralized network management, because it prevents redundancy in gathering and processing network management information. It also allows for an uncoordinated way of achieving similar management goals compared to a coordinated approach to decentralized network management. The probabilistic behavior makes the framework particularly attractive for the management of highly dynamic network environments, such as mobile and ad-hoc networks. In those environments, the uncoordinated property is specifically beneficial.

Still, probabilistic management must achieve similar results as conventional network management; therefore, we evaluate our paradigm in the context of a network of information (NetInf) [2], a clean-slate approach towards an information-centric future Internet. Rather than addressing nodes, NetInf addresses information elements in the network.

Manuscript received August 8, 2008. This work was supported in part by the European Union through the 4WARD project (http://www.4ward-project.eu/) in the 7th Framework Programme. The views expressed in this paper are solely those of the authors and do not necessarily represent the views of their employers, the 4WARD project, or the Commission of the European Union.

After a brief introduction of the in-network management framework in Section II, we describe probabilistic management in Section III. We evaluate the concept in the context of NetInf in Section IV and discuss the results and the applicability of the probabilistic paradigm to real environments in Section V. We close with a differentiation of our work and the state of the art (Section VI) and summarize the approach including future work in Section VII.

II. Background: In-Network Management

In traditional Internet management (Fig. 1, left), functionality resides outside the network in management stations and servers. These entities interact via management protocols such as SNMP or CLI with the network elements to execute FCAPS management functions, such as fault and performance management. In commercial networks, these interactions often occur out-of-band through special communication networks.

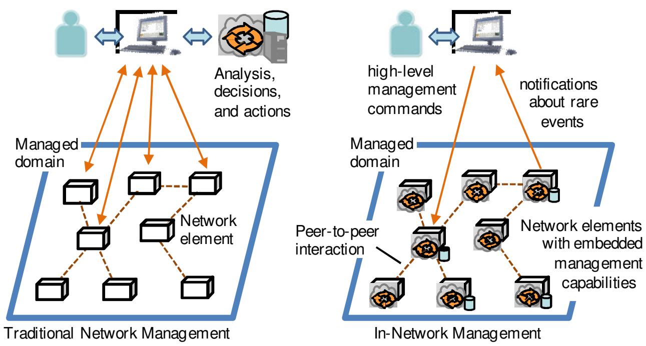

Fig. 1. Traditional and in-network management.

Such a management approach has proven successful for relatively small networks (up to a few hundreds of nodes) and static configurations. For emerging large-scale, dynamic, and heterogeneous network environments, however, this approach turns out to be inadequate. This situation has been recognized and attempts to decentralized management have been proposed. For instance, SNMPv2 [3] introduces the concept of intermediary managers that interact in a weakly decentralized hierarchical structure for distributed data collection. Another example is RMON [4], which uses monitors and probes that allow making decisions outside the agent and local to an occurring anomaly. However, such approaches are highly limited in their functionality and are far from reflecting the inherent nature of management to networks.

To overcome the limitations of current management technologies, we pursue in the scope of the 4WARD project [5][6] a new paradigm for network management, which we call *in-network management* (INM) [7]. INM's key idea is that management processes are implemented as embedded management capabilities inside of network nodes, forming a self-organizing and inherent management plane (Fig. 1, right) that requires only the bare minimum of human intervention. INM provides the principles, methodology, and framework for implementing embedded management functions in current and future communication networks.

III. Probabilistic Management Framework

Since management functions are in many cases redundant across the network, we propose to turn them on or off randomly. At one point in time, thus, some functions are turned on, while others are not. The specific way of how the activation of management functions is realized may depend on additional system constraints and performance tradeoffs. If all functions that are subject to the probabilistic management process are instantiated and resident in the memory of the networked device beforehand, rapid switching between on and off states is possible, which allows fine-granular probabilistic control. It is also conceivable to include the installing and uninstalling of management functions in the randomization process, for instance, if constraints in the transient memory of the networking device apply. While memory resources are conserved, CPU utilization increases and more coarse-grained turning on and off is more advisable. In general, different tradeoffs between the cost of activating and deactivating a function and resource savings are possible. For ease of discussion and without loss of generality, we assume in the following the first of the discussed alternatives.

By control and management functions, we refer to classical functions including, but not limiting to, fault, performance, and configuration management. Note that by "random", we refer to any type of randomness, including pseudorandom and perfectly random processes. While perfect randomness is difficult to achieve with today's technology, pseudorandom behavior, implemented by popular random number generators, is absolutely sufficient for our applications.

A. Framework Overview

Fig. 2 illustrates the basic layout of our probabilistic management framework. A set of management functions is running on the node, interacting with the networking functionality directly or through the node's database(s) or information store(s). A component called the Randomization Process designates the meta-management entity that takes care of randomization of the management functionality. The randomization process can be influenced through various factors and might be configured from an external entity.

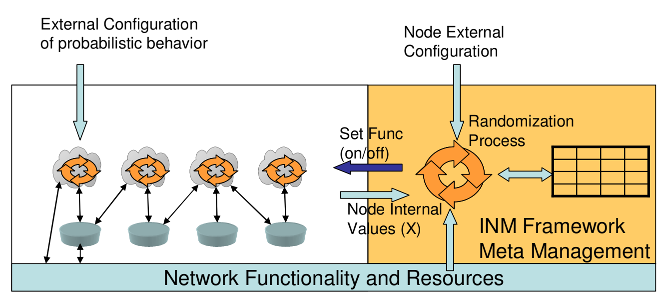

Fig. 2. Basic probabilistic management framework (node view).

The randomization process on the network element has a probability function f_i with a certain probability distribution per management function on the node (Fig. 3). An interval I_i denotes the interval between two successive executions of the randomization process for a certain function, which can be fixed, dynamic, or random. Each time the interval I_i elapses, function f_i decides whether or not a function is turned on or off. Both probability function and the interval may depend on configuration, type of function as well as configuration and internal information. In a special case, those values might also depend on information about neighboring nodes. However, such models introduce additional, more complicated dependencies again, which we attempt to minimize with the probabilistic scheme in the first place.

Function	Interval	Probability Function	Parameters
f_1	I_1	Uniformly distributed within range	Low = 1 High = 10
f_2	I_2	Uniformly distributed with average	Avg = 0.2
f_3	I_3	Normally distributed with average	Avg = *f* (node-internal value X)
f_4	I_4	Equally distributed with average	Avg = *f* (node capability)
…	…	…	…

Fig. 3. Random process configuration table.

The probability distribution can be adapted in dependence of information that is either internal or external to a node, or both. A typical example of adaptation may occur based on the node-internal information of free memory capacity. The probability distribution may then correlate to the remaining memory in that the likelihood of turning the function on and off increases and decreases, respectively, with free memory capacity. For functions requiring dedicated memory for information storage while running, such as measurement and monitoring functions, the likelihood of those functions being active depends on the locally available storage. Typically, the probability distribution may be distributed exponentially, meaning that as long as the storage is fairly empty, the likelihood to run the function is very high, while after a certain point the likelihood is increasing rapidly.

B. *Probabilistic Management Interfaces*

The interfaces defined by the probabilistic management framework pertain to the setting, deleting, and changing of the random process configuration table for each function. This means that a function must be identified uniquely within the node. Given that the same management function may run multiple times in a separate instance, each instance shows up as a separate function in the configuration table.

The interface for receiving node-internal information from either the management functions or directly from the node's network functionality and resource management functions is not further specified. The reception of information in both pull and push model depend on the instrumentation of the components that send the information.

The interface of management functions basically includes the setting of the management function to the on or off state. Since setting the function on includes a certain instantiation or installation step, a second interface supports the notification of the function that it is being turned on. This differentiation allows for more independence from the specific node-internal execution environment. For example, in our Java-based simulation prototype in Section IV, we instantiate each management function as an object, which is then notified through a method call about its activation. Conversely, when the function is turned off, it is not deinstalled, but rather notified about its being deactivated. The process of finding management functions and including the knowledge about each function is a basic feature of the in-network management framework, and is here not further discussed.

C. *Interdependency with Random Function Behavior*

Besides the introduced dedicated randomization process, the management functions themselves may have an internal probabilistic behavior. Such functions therefore support an interface to configure function-specific probability-related distributions or configuration settings.

A representative example for such interdependency is the case of nodes monitoring their neighbors for a failing or misbehaving node. Let us assume that the management function defines an internal probability value, say 0.3. Hence, with the probability of 0.3, a node selects any of its neighbors to monitor and actively check against failures through pinging other neighbors. In that case, the information on how many neighbors do monitor a node in the mean is relevant for deciding the probability of a node to set a neighbor monitoring function on or off. Such scenarios require the cooperation between the randomization process of the probabilistic framework and the management function itself.

This type of interaction, however, should be considered the rare case, because it introduces deterministic behavior that is contrary to the motivation of probabilistic behavior. Assuming that a purely deterministic algorithm can be readily used, the reduction of resources, for instance, can be achieved more easily and in a well-controlled way by means of aggregation. For example, let us assume that a node requires two neighbors to participate in a monitoring task. In this case, the node simply tells all its neighbors how many nodes it still needs and how many monitors it already has. Instead of a randomization process, an explicit coordination algorithm can be used.

While such an approach works in the previous example, it is generally not feasible in a network with complex function types and particular implementations that might not be compatible in terms of coordination or cooperation. Furthermore, coordination introduces additional communication overhead, which may be significant for a large number of functions executing in parallel.

D. *Influencing the Probability*

In general the adaptation of the probabilistic mechanisms is performed through setting of probabilities, distributions, or parameters that depend on function type, system and network environment, operational requirements, and activity level of the management function. In the latter case, when a function uses a large part of the CPU time without reading or writing any values, the assumption is that this function can run less often. The probability that such a function is active is then set to a lower value. For example, in Fig. 4, if the relationship between CPU usage and configuration actions (setting values in the system) is not in relation, a change of such a function's probability to run can be foreseen. The topmost function f_4 is using 25% of the CPU time, but does not have any write operation on values for configuration. Therefore, the probability of function f_4 being executed is decreased, which in the long run (i.e., not in timeframes of the decision making process), lowers the usage of the CPU. Note that this does not imply anything about short-time function scheduling, but that the function might run less frequently in the future.

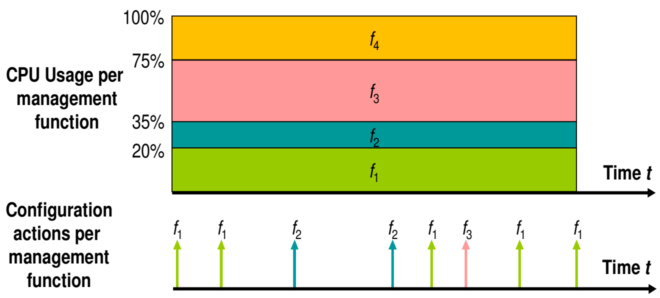

Fig. 4. Dependency of resource usage and activity level.

The probability to turn on and off a function might also depend on the function's additional characteristics. For instance, when management functionality is operationally critical, the implementing function must be executed with high probability. Conversely, less important functionality can be set to a smaller execution probability. For achieving a certain management goal, the probability function and parameters might need to be changed from externally. For example, in order to increase the accuracy of management information that a function exports, the execution probability of that function has to be increased in the first place (cf. Section IV.B).

For setting of the probability variables, the control loop can go through a centralized component reading or writing values in the management function on the node. For example, when the monitored values calculated by the management function are fairly similar across all functions of the same type on different nodes, there is a good chance that those management functions are very redundant, and the probability of that function to be run can be decreased for the future.

IV. Evaluation

We have implemented parts of the INM framework in a Java-based simulation environment and added functionality for creating and running probabilistic decentralized management scenarios. We are able to dynamically add management functionality to a node, which is in turn automatically added to the randomization process handling the respective function. The probability distributions can be parameterized dynamically by setting a specific random number generator in the randomization table.

A. *Application to a Network of Information*

We apply the probabilistic management framework to a future Internet approach called the *Network of Information* (NetInf) [2]. Motivated by the fact that users are more interested in the information, rather than the individual nodes storing the information, NetInf defines a new information-centric paradigm. Rather than building on the networking paradigm of node-centric communication, NetInf exploits information-centricity to connect and relate information elements with one another and to directly build dictionary and management structures on top of these elements. The essence for our purposes is that NetInf provides a global distributed information store, where applications publish information and are able to query information from the system.

Fig. 5 illustrates the basic NetInf architecture. The left side shows the network structure, which runs on top of today's IP networks, but may also run on top of novel future Internet technologies to be developed. The right side shows a sketch of a node's partial internal structure, including the NetInf-specific functionality and the probabilistic management framework. Application access the system through an API on top, and the system uses some networking technology at the bottom. The dots in the different functions of the network of information system (e.g. caching/storage) denote management functionality associated with those service-specific functionalities.

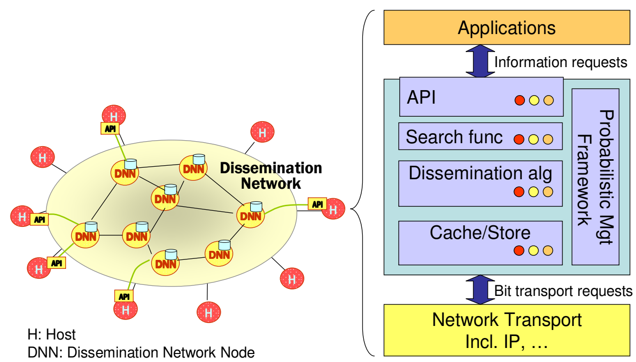

Fig. 5. Network of Information (NetInf).

The management information of the NetInf system that we use in the following comprises the number of API calls, the size of each cache, and the number of transport requests. We will use more of the internal NetInf functionalities later on, once we will have introduced more specific mechanisms and algorithms of the network of information.

B. *Results*

For all of the following simulations we used the home-grown simulator implementation based on Java. We executed 20 simulation runs per data point in a scenario of 100 nodes. All nodes are dissemination nodes, sending information objects when requested by an application node. Every node is an application node as well, hosting an application emulator pushing information into the system as well as requesting information elements. We assume dissemination on demand, meaning that information objects are sent to a new node only when requested by that node. Once an information object is delivered to that node, it remains in that node's local data cache and may be used for future requests of other nodes. The objects to be retrieved are randomly chosen with an equal distribution over all existing information objects in the system.

The interval of publishing and retrieval requests is equally distributed over a fixed range. There are ten times more information requests than object publications. Finally, we assume a fixed simulation time of 1000 seconds.

1) Monitoring of the Network of Information

In this scenario, we study the effects of probabilistic management in a well-balanced setting in order to understand the effects of the approach in common situations. In the simulations we monitored various values including the number of API requests for retrieving and publishing information, internal cache size, number of transport requests etc. We divided the overall monitoring task into two functions, each of which can be turned on or off randomly. One of the functions monitors the API requests from applications; the other reads data that is gathered in the NetInf system anyway for internal use.

We considered the value of the number of API information retrieval requests. When extrapolating the monitored values to the overall network and monitoring time, we have the same average number of information requests per node independent of the probability of running the monitoring function. The average amount of data gathered per node, however naturally decreases with smaller probabilities (Fig. 6), therefore less monitoring instances are running in the network. This is as expected in a system with equally distributed activity.

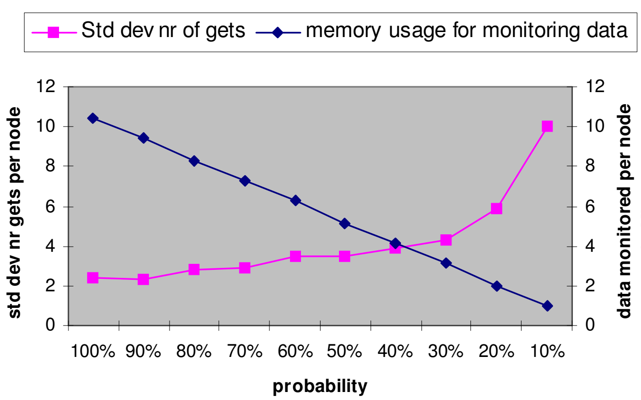

Fig. 6. Standard deviation and amount of monitoring data gathered.

However, as shown in Fig. 6, the standard deviation across all simulation runs differs with the probability of running the monitoring function. This means that the extrapolated data's accuracy is smaller than when all monitored data is considered. We observe, however, that down to a probability of just 0.3, which is equivalent to the running of only about one third of the monitoring functions, the standard deviation only changes insignificantly (see Fig 7 for the same numbers represented as percentages). This underpins the ability of probabilistic methods to achieve accuracy levels that are similar to ideal methods in the scope of network management. Note that the standard deviation is not zero since the there is also variability through the random generation of the network load.

So far we have chosen a fairly balanced load model for the network of information. In the following scenario, we modify the load model to have ten dedicated nodes in the network that do a lot of information publishing. In average, they do the same amount of publishing as other nodes do retrievals, but far less retrievals than the other nodes.

Fig. 7 shows the comparison of the standard deviations in the case of balanced and unbalanced load. In an unbalanced setting, the deviation increases faster and reaches a larger value than in a balanced setting. Hence, the monitoring accuracy is smaller. Still the average number of retrievals and the average number of publishes, when extrapolated, are the same no matter what probability we have chosen. Also the monitoring data size is the same as shown in Fig. 6. Although in the unbalanced case, the standard deviation grows significantly, it still remains below four percent at a probability of 0.3. This is a remarkably low standard deviation, which means that even in unbalanced scenarios, probability-based methods allow significant resource savings while retaining a high level of the monitored information's accuracy.

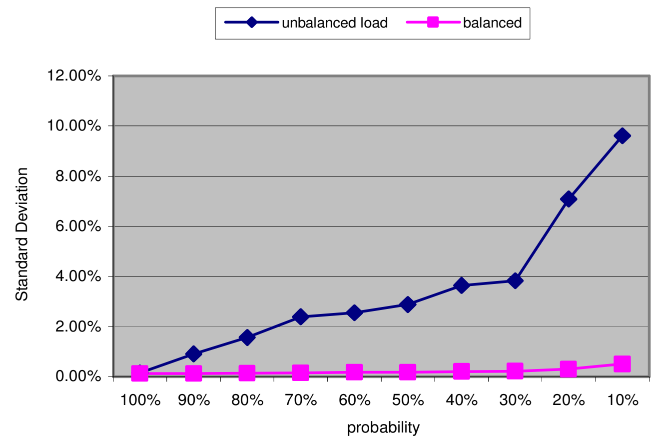

Fig. 7. Standard deviation of balanced vs. unbalanced load.

2) Fault Management in the Network of Information

The next set of simulations is concerned with fault management functions. We define a fault as the situation where a node is down and hence, unreachable for responding to object information retrieval requests. We detect a node failure when another node attempts to retrieve an object from the failed node, but was not able to do so. We do not differentiate whether such a failure is due to a network or a node problem. Note that there are several different ways of doing fault management. For simplicity, we restrict the following discussions to only one possible way that we use to analyze the suitability of the probabilistic management.

For the following simulations we continue to use the previously stated NetInf setting. We randomly choose 100 nodes which at a random time during the simulation fail for a duration of 2 seconds. We have chosen this value because it denotes the boundary where failed nodes are showing up in the form of information retrieval errors. If the failure duration is shorter, there is a set of failed nodes that are not detected, since no information retrieval requests destined for that node fall within the failure window. However, Fig. 8 shows that also for a probability of 100% to run the fault management functionality, a small number of node failures are still not detected by the NetInf. In order to also detect these failures, targeted fault management would be required that actively checks the nodes' mutual reachability.

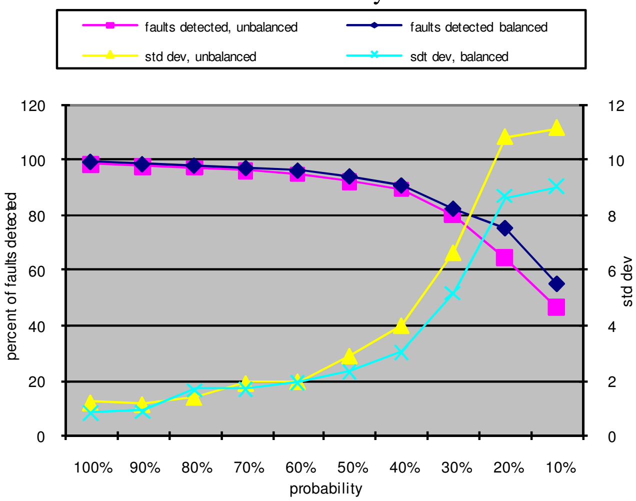

Fig. 8. Fault detection.

Regarding the probability-based detection of faults, Fig. 8 shows that the fraction of detected faults is fairly small down to a probability of approximately 50%. This observation is true for both the balanced and unbalanced setting, which are both shown in the figure. Furthermore, even at a probability of only 10%, it is still possible to detect about half of the occurring failures. Note that the resource usage decreases linearly with the probability to run a management function.

Comparing the balanced and unbalanced case, the latter shows only a slight decrease in the ability to detect node faults and the standard deviation thereof. This confirms that the probabilistic management paradigm is not only applicable for homogeneous scenarios. It also tolerates inhomogeneous load models and is able to achieve high accuracy.

V. Discussion

A. How can the probabilistic management system be supported in current and future Internet architectures?

In order to support our probabilistic management framework, we have identified the following five key requirements based on the discussions in Section III:

Granularity: Network management functions should be built into the network in a fine-grained way, so that turning on and off can occur on a detailed level (cf. Fig. 5).

Overhead: The probabilistic control logic shall be light-weight in nature, in the sense that turning management functions on and off involves only low overhead.

Flexibility: It should be possible to customize the way in which probabilistic methods are applied. For instance, it is desirable to easily accommodate additional management functions into randomization processes.

Transitivity: Because management functions naturally interact within control loops, turning on or off single functions may not imply the desired resource savings. For example, in the distributed aggregation of performance values, turning off top-level reporting at the aggregation root might still lead to updates within the aggregation tree. Therefore, the turning on and off process may also require the consistent (de)activation of transitively connected management functions.

Interactivity: Despite the fact that management functions are embedded into the network, it is still necessary to interact with management functions from externally, for example, to tune and influence probabilistic processes based on additional external knowledge (cf. Fig. 2).

In order to support probabilistic decentralized network management under the given requirements, INM provides two architectural elements: self-managing functional components (FCs) and fine-grained management capabilities (MCs).

Fig. 9 shows two FCs that implement the *Search func* and *Dissemination alg* layer in Fig. 5. Each FC contains two planes, denoted integrated and inherent management plane. Inherent management pertains to management functions that are inseparable from and shipped with the FC. For example, a dissemination algorithm may manage its own set of parameters to tune dissemination performance. Integrated management refers to modular management functionality that exports interfaces to other management functions and which can, optionally, be added to or removed from an FC. The probabilistic management functionality can be modeled by an integrated management function (cf. Fig. 5 and Fig. 9).

Integrated management functions publish their functionality via the integrated and external management interfaces. To the latter, only high-level management interactions are published (e.g. related to service management), while only low-level ones are published to the former (e.g. related to network element management). The clear separation is motivated by the fact that it is desired to reduce human interaction to only high-level objectives (e.g. business goals).

All management functionality is implemented via fine-grained MCs, thus naturally supporting the granularity requirement. Each MC exports specific interfaces. Some MCs interact with the inherent management plane through cross-plane invocation, thereby connecting inherent and integrated management. This is useful, e.g., for monitoring performance parameters that are inherent to an FC. If desired, any MC may implement a probabilistic interface to connect to the probabilistic MC, thereby plugging into the probabilistic management framework according to Fig. 2.

The process of capability embedding directly supports the overhead requirement. Firstly, probabilistic management is implemented as a light-weight MC itself. The switching on and off then reduces, for instance, to a function call to other MCs by the probabilistic MC (e.g. within a Linux kernel module, or a sensor node's monolithic code block).

Support for the flexibility requirement is an integrated feature of the INM framework, which allows the adding and removal of MCs. Due to the light-weight nature of the MCs, these processes are also light-weight.

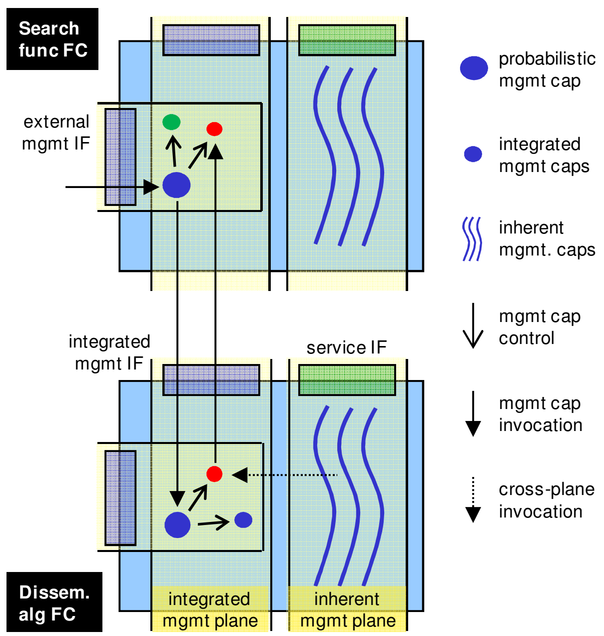

Fig. 9. Structure of and interactions between functional components.

The transitivity requirement is supported in two ways. Assuming an MC implements a probabilistic interface as noted above, the MC can support transitivity directly, e.g., when the MC knows best how to inhibit other transitively connected MCs. On the other side, the probabilistic framework may take the role of transitively turn on and off MCs shown in Fig. 9 by an interaction between the probabilistic MCs in both FCs. This works, e.g., in such cases where the governed turning on and off does not impact the semantics of a management function.

Finally, the interactivity requirement is supported by the external management interface, which allows external parties to interact with the probabilistic MC, e.g., to set new probability values (which might in turn be propagated to an MC via its probabilistic interface, cf. Fig. 2).

Additional details on the INM framework are given in [7].

B. Are the numbers we obtain from a probabilistic management system good enough for managing a network?

As shown in Fig. 6, as long as the system is well balanced, we can run the probabilistic management system on very low probabilities and get an error no larger than 0.2%. Even when reducing the probability down to as low as 30%, which in turn means a reduction of monitoring traffic by 70%, no accuracy is sacrificed for average values across the network.

In more unbalanced scenarios, the extrapolation of values is less accurate than in balanced ones, and naturally accuracy of the observed management data decreases with smaller probability values. In the presented case of fault management, we are still able to achieve a 95% success rate in the detection of faults. This value is combined from both probabilistic management and from the effect that no service requests at all occur at a subset of the faulty nodes.

The difference between balanced and unbalanced scenarios is smaller for fault management than for the monitoring case. Since fault management may be more critical in some scenarios than just monitoring, our results demonstrate that even more critical management tasks are suitable for being subjected to probabilistic management.

C. Is the concept of probabilistic management acceptable by users and operators?

In general, probabilistic systems lack acceptability due to their nature of being not intuitively understandable. Specifically, looking into control and management systems, there is a fairly large resistance to systems that incorporate a certain degree of randomness. If we assume that the decentralized self-managing design of future network management will become reality, we will already have a certain degree of uncertainty in the self-managing system itself, doing configuration and performance management automatically. We feel that in such situations a certain degree of randomness does not harm.

More interestingly, many of today's successful networking technologies do depend on significant degrees of randomness. One of the most prominent examples is Ethernet's highly successful CSMA/CD scheme, which employs randomization for solving distributed medium access problems. Another example is statistical multiplexing performed in IP networks. Possibly the most striking fact is that even in network security, randomization is a widely accepted and applied technique that lies at the basis of many cryptographic protocols. All of these scenarios share that the underlying systems contain certain degrees of nondeterministic behavior, which does not allow the application of purely deterministic mechanisms. While these randomization processes are sometimes difficult to grasp, we believe that, based on our evaluation, they will also be vital in future communication networks, which will be characterized by significant complexity where indeterminacy and unpredictability will play a major role.

On the business side, it is more relevant to assess the commercial benefit of lowering the resource usage for network management and paying with less precision in some cases. It always makes sense to design resource-efficient systems, specifically, in mobile and resource-limited environments. In many novel network architectures, the management of very dynamic networks like car-to-car communication, ad-hoc networks and peer-to-peer is required. In such environments, probabilistic management helps by easing the introduction of management into the dynamic system, which would not be possible with traditional management paradigms.

VI. Related Work

In the area of traffic measurement, the idea of packet sampling [8][9] has been used for reducing the amount of management information or the reduction of processing needs for high-speed link packet capturing. In order to enable capturing packets on high speed links with low cost, only every n-th packet is captured or exported. Our approach does not deal with a specific management function like measurements, but rather turns on and off randomly the complete function. We do for the moment not assume any randomization within the function itself, which is function-specific and therefore out of our control.

A second area of related work is that of scheduling, specifically of process scheduling [10]. Assuming that our functions are running within process, we could argue we randomly set processes to sleep. In principle, our work has little to do with classical process scheduling, since the job or task is not even there to be scheduled, when it is turned off. So no resources are used at all. Also we do not optimize the resource scheduling. Rather we achieve efficiency through not running unnecessary many of the same functions on the network for gaining the same or similar results.

The third area of related mechanisms is the class of randomness in media access protocols [11]. For example, randomized scheduling for medium access, both wired and wireless, is a known way of removing a deterministic coordination function for decentralized medium access system. In our case, we use a similar approach applied to decentralized network management functionality.

There is quite some work in sensor and ad-hoc network management [12][13]. However such work does not take into account the approach of probabilistic management, but rather deal with the problem of low resource management paradigms to lower the cost in resource-limited networking environments. Work being in the same area, but specifically using also probabilistic behavior is the following. [14] proposes to use the connectivity probability of ad-hoc nodes in order to define clusters reporting to a management node. Thus, the management plane connectivity is based on a probabilistic behavior. We go a step further and run certain functionality on a node in a probabilistic way. However, we can naturally take the connectivity probability into account as an input to derive the run probability of a function (dependent on the function itself). The authors of [15] propose a self-managing system for ad-hoc network management, which spreads management functionality across different nodes taking into account the nodes' capabilities in terms of resources. In our probabilistic framework, the capability would influence the probability of running a certain function on a certain node.

VII. Summary and Future Work

In this paper, we proposed a fairly generic probabilistic management framework for efficiently managing networks in a decentralized way. The probabilistic management paradigm allows for a generic mechanism to turn on or off certain management functionally based on a probability, which can be influenced by the operator, the capabilities of the node, the function itself, or any other external information.

So far we have only scratched the surface of the potential of such a paradigm. We believe that in many cases it is useful to do without deterministic management behavior. While this is against the traditional thinking of telecommunications or internet operators and administrators, we need to consider how to make them comfortable with such a paradigm. For example, we can introduce decentralized management with a probability of 100%, showing that the system correctly works for some functions, and only then slowly reduce the percentages of running the functions. The resource usage reduction could then be used for running other functions, both management or non-management functions alike.

Finally, there are naturally certain application areas, which do not profit or where it is not feasible to run such a probabilistic management framework, such as real-time or highly reliable systems. Those must be identified and optimally, classified based on a suitable taxonomy, in order to have clear application and use cases where probabilistic management works and where it doesn't. This also means that we need to extend and evaluate the paradigm in other networking systems than the network of information.

References

[1] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, J. Turner, "OpenFlow: enabling innovation in campus networks", *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69-74, Apr. 2008.

[2] C. Dannewitz, K. Pentikousis, R. Rembarz, É. Renault, O. Strandberg, and J. Ubillos, „Scenarios and research issues for a network of information," *Proc. MobiMedia'08*, Oulu, Finland, Jul. 2008.

[3] Internet Engineering Task Force (IETF), "Introduction to Version 2 of the Internet-Standard Network Management Framework," *Request for Comments (RFC)* 1441, April 1993.

[4] Internet Engineering Task Force (IETF): "Remote Network Monitoring Management Information Base," *RFC 2819*, May 2000.

[5] N. Niebert, S. Baucke, I. El-Khayat, M. Johnsson, B. Ohlman, H. Abramowicz, K. Wuenstel, H. Woesner, J. Quittek, and L. M. Correia, "The way 4WARD to the creation of a future Internet," *Proc. PIMRC'08*, Cannes, France, Sep. 2008. To appear.

[6] "4WARD: Architecture and Design for the Future Internet," Collaborative Research Project within the European Commission 7th Framework Programme (FP7), online: http://www.4ward-project.eu/.

[7] C. Foley, S. Balasubramaniam, E. Power, M. P. de Leon, D. Botvich, D. Dudkowski, G. Nunzi and C. Mingardi, "A framework for in-network management in heterogeneous future communication networks," *Proc. MACE'08*, Samos Island, Greece, Sep. 2008. To appear.

[8] Nick Duffield, ed., "A Framework for packet selection and reporting," Internet Draft, draft-ietf-psamp-framework-13.txt, June 2008.

[9] K. C. Claffy, G. C. Polyzos, and H.-W. Braun, "Application of sampling methodologies to network traffic characterization," *ACM SIGCOMM Computer Comm. Review*, vol. 23, no. 4, pp. 194-203, Oct. 1993.

[10] J. Nino-Mora, "Stochastic scheduling," *Encyclopedia of Optimization*, vol. V, pp. 367-372, 2001. Kluwer. Updated version.

[11] G. Mergen and L. Tong, "Random scheduling medium access for wireless ad hoc networks," *Proc. MILCOM'02*, vol. 2, pp. 868-872, Anaheim, California, USA, Oct. 2002.

[12] W. Chen, N. Jain, and S. Singh, "ANMP: Ad-Hoc Network Management Protocol," *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 8, pp. 1506-1531, Aug. 1999.

[13] L. B. Ruiz, J. M. Nogueira, and A. A. F. Loureiro, "MANNA: A management architecture for wireless sensor networks," *IEEE Communications Magazine*, vol. 41, no. 2, pp. 116-125, Feb. 2003.

[14] R. Badonnel, R. State, and O. Festor, "Probabilistic management of ad-hoc networks," Proc. *NOMS'06*, p. 339-350, Vancouver, Canada, Apr. 2006.

[15] C.-C. Shen, C. Jaikaeo, C. Srisathapornphat, and Z. Huang, "The Guerilla management architecture for ad-hoc networks," Proc. *MILCOM'02*, vol. 1, pp. 467-472, Anaheim, California, USA, Oct. 2002.