Opportunities, Requirements and Challenges for Storing Network Management Information in a Decentralized Way

Marcus Brunner, Frank-Uwe Andersen

***Abstract*—The upcoming peer-to-peer (P2P) and other decentralized, co-operative storage mechanisms allow for decentralized storage of data as well as decentralized search, depending on the specific system. In this work, we assume these kinds of systems to store management information into it. We discuss some of the opportunities, requirements, and challenges when storing network management information decentralized. Additionally, we discuss some issue, when we assume the management plane itself being decentralized, adding the aspect of distributed processing. Distribution of storage and processing resources make the use of such systems very useful, but still require some changes from today's known systems, which are typically targeted to store content for end-users, for example in file-sharing or content delivery networks (CDN).**

***Index Terms*—Decentralized Management, Peer-to-peer Information Management, Distributed Networking**

I. Introduction

THIS paper addresses issues around network management information storage and retrieval in decentralized settings. We consider decentralization of both the information storage as well as the network management functionality. Both of which traditionally work in a very centralized and hierarchical way in telecommunication systems, in line with the paradigmatic TMF pyramid, where network elements are handled by element managers, which are in turn controlled by a network manager, and even higher layers for service and business management. Note that many of the discussion would apply similarly to distributed IT systems management, but we do not further detail those.

While we are not claiming that a complete, overall change towards a fully distributed NM system is generally necessary, we believe that it is justified to rethink and compare alternative structures and technologies that enable partial or even completely distributed designs or solutions, especially if benefits can be identified when comparing with traditional network management systems and procedures. Specifically, we assume decentralized storage, but the processing of network management functions can be central or decentral.

Specifically, we are referring to information storage and handling procedures currently being researched in EU FP7 ICT 4WARD [1], which are based on separating content and locator information and one or multiple indirection steps, resulting in the ability to store and retrieve information objects in a more generic yet This paradigm is called the Network of Information (NoI), and it is foreseen to provide a storage middleware allowing to store information elements in a decentralized way and the search and retrieval of those. It can be seen as a generalization of various systems in the P2P, CDN, and file sharing space. Since this particular system is under design at the moment, we try to find out what the requirements and challenges are, when we want to use the system for storing network management information in it, and retrieve that data from it.

The underlying problems can be formulated as follows:

- When we assume decentralized management functions, they invite or even call for decentralized (or local) storage to operate on.
- When we assume to keep network management information within the network, it requires more intelligent access and retrieval mechanisms
- In any network, the network management information source is distributed across the network. Therefore, the decentralized storage might better adapt to the decentralized nature of the information. Specifically, when the information might be needed locally on the network element, or at various places in the network or at a central station.

The expected improvements include:

- Decentralized data storage has benefits in reliability and local access speed. We want to benefit from those properties for network management.

The main challenges are:

- Typically, NoI systems are not targeted to network management information. What is really needed to make them useful for storing network management information? We do not assume any specific decentralized data storage system itself, but they finding should be applicable to most of them including the one under development in the 4ward project.
- We might need to re-formulate network management

Manuscript received August 6, 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.

M. Brunner, is with the Network Laboratories of NEC Europe Ltd., Heidelberg, Germany. brunner@nw.neclab.eu

F.-U. Andersen is with Nokia-Siemens Networks, Berlin, Germany, frank-uwe.andersen@nsn.com

information flows and procedures (a) to be in line with NoI system (b) to exploit the full potential of NoI system.

- Many of the search and retrival mechnisms in NoI systems are built for a specific purpose, and might be difficult to adapt to the use in network management. .

II. Background on Decentralized Data Storage

A. *Peer-to-peer and DHTs*

Typically, DHTs or other Peer-to-peer paradigm-based system do store information decentralized and find information in a decentralized way, and can even retrieve it in a distributed fashion (multi-source parallel download). The structure and topology of the network as well as the routing algorithm (especially the DHT metric) towards the information differ between several such systems (Kademlia, CAN, Pastry, …). In this paper, we will not suggest the use of a particular DHT; instead we try to convey the idea of using a new, distributed information storage concept, called the "NoI" (introduced in the previous section and further detailed in the next section) for the storage of network management related information.

B. *Network of Information*

The overall objective for a NoI [3] is to design a general, information-centric network architecture, which is concerned with information retrieval and storage. It is concerned with the information objects themselves rather than the nodes that host them. Information objects are directly addressed, without any knowledge of or on what node they are actually are hosted. The main components of a NoI are a modeling framework that facilitates object discovery and use on the basis of their names, and a reference model specifying the syntax and semantics of object operations. In addition, specific networking services and mechanisms are used within the architecture. The architecture is based on two boundaries, a lower API towards the network infrastructure, such as IP, and an upper level API, which is meant to be used by future applications. In our case we basically assume that the management functionality uses the upper API for the storage as well as the retrieval of management information.

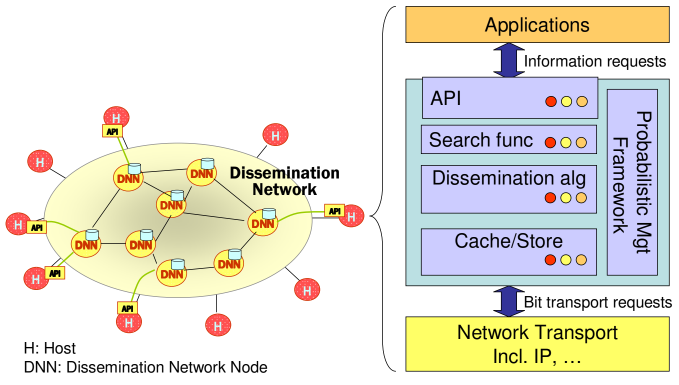

Figure 1: NoI Overview

Most peer-to-peer systems are only used for simple non-wildcarded searching of information rather than storing the information itself. Also many do depend on DHTs in their core. However, that depends on the specific application it is used for. And it is unclear in general, what the right approach really is, and how much data handling is required in the decentralized storage system. The NoI system is foreseen to provide a larger breadth of functionality including decentralized storage, but also search and retrieval functionality.

The NoI system under development at the moment tries to cover a broader space, and therefore, the usage for network management might be one use case to consider in the design of such a novel system.

For the rest of the paper, we assume the decentralized storage system as a blackbox with the capability to put information in and to retrieve information from the system and discuss on a high level the issues and requirements, when such a system should be useful for network management information storage.

III. The Path to Decentralized, In-Network Management and Related Work

Traditionally, the network management (NM) is logically and also location-wise fairly centralized, and all the management related information is retrieved from the network through some management protocols. In the central location, the management information is typically stored in a central database. Also the network management system is in charge of retrieving the information from the network, processing it and storing it in the database. Another characteristic is the hierarchy in traditional NM; every network element (NE) has a relationship with an element manager (EM), which in turn is connected to the overall, topmost NM system.

The first set of enhancement had been the decoupling of the network management functionality from the retrieval and setting processes. Most prominent work has been done by J. Strassner [5] in the context of Directory-enabled Networking (DEN-ng). Still, the storage is central, but accessible remotely from any network management application, typically through LDAP or any other protocols.

On the other hand, the way towards decentralized management has been paved through various work on Management by Delegation [7], the IETF Script MIB[8], and patterns for decentralized management [6] increasing the degree of decentralization.

Madeira [9] suggest to use the "P2P principles" for distributed network management and follow a model-driven approach. They do not provide details on storage procedures, however.

As a next step, we propose to go even further into the network with the network management functionality, not only decentralizing the functionality. We call it In-Network Management (INM), a different paradigm for network management [1][2], where management functions come as embedded capabilities of the devices. With this approach, network elements have embedded "default-on" management capabilities, consisting of several autonomous components which co-operatively interact with each other in the same device and with components in neighbouring devices. Glued together with a set of discovery and self-organizing algorithms, the network elements form a thin "management plane" embedded in the network itself.

The In-Network Management paradigm can be interpreted as pushing management intelligence into the network, and, as a consequence, making the network more intelligent: as a consequence, objectives and costs of management operations can be adapted according to local working conditions. The network, which now inherently includes the management plane as a part of itself, can execute end-to-end management functions on its own and perform, for instance, reconfigurations in an autonomous fashion. It reports results of management actions to an external management system, and it triggers alarms if intervention from outside is needed.

IV. Storing Management Information as information objects

How can the concept of information objects, consisting of indirection (i.e. naming/addressing mapping or resolution) provided by NoI, and storage/retrieval be used for decentralized, in-network management? NoI allows creating more or less persistent information objects via an API, which are then accessible via an indirection and resolution mechanism. For the storage part, two options are possible: The network management or network state information is stored either on the network nodes that produced them, and in that case, NoI is only used to create a NoI representation of the real object. Or, the information object is itself handed over to NoI , which then additionally to providing the mappings takes care of the storage task, on nodes that it determines itself. For retrieval of the information object(s), it is also possible that a only the search of the node where the objects are host is part of the NoI, or that dedicated transport mechnisms is used to get the data object to the requested place in the network.

In the preceding section, we already have suggested to separate the data handling from the management part and procedures. It makes sense to sub-group the types of data that usually occur in network management. The typical network management control loop involves data that could be seen as separated into (at least) three groups for our purpose. Separation of the overall NM data can help to analyze how and where the concept of information objects can be applied in a beneficial way.

In a broad sense, the separation of NM data can be made into (1) control commands, which are many times just modeled as information objects, but could get made available with other mechnisms (not availbele in the traditional network mangmeent protocols). (2) measurement data about what happens on network elements. (3)s tate information, which is fairly local, but could potentially be relevant for others (known also as "triggers", "notifications", "events" or "alerts", depending on further classification).

If we consider that the NoI API (see preceding section) offers two basic types of services that can be used by NM, we can arrange them together with the NM data types in one table in order to see which services apply to which data types, and in which way. The first one is searching for objects according to certain criteria. The second one is storage/retrieval of found or known objects. There is a third one, however, and this is the mapping or resolution of objects IDs onto physical (routable) locations. It is part of the storage/retrieval service but worth to be recognized as a separate service that could be considered the future Internet's "DNS".

Table 1: NM data types and NoI methods

	Search / Retrieve	Storage (object creation)	Resolution / Mapping
Control comman ds	Determine nodes to control based on certain criteria	(certain control procedures might be stored as objects as traditionally done)	Determine physical location of one or more target nodes
Measure-ment data	Determine nodes to exchange measurements with or search for specific measurments on a set of nodes	Aggregated measurement data	
State informati on	Determine nodes to send triggers to and store triggering information within NoI objects.	Aggregated state information	

Regarding NM control commands, it is imaginable for nodes that issue control commands to other nodes to use an object ID as an address. The object ID would represent one or more target nodes, and it is the task of the NoI system to resolve this. There is certainly more flexibility in this approach compared to the relatively simple domain name on IP address mapping that DNS (or even DynDNS) can offer. The resolution process for a NoI information can potentially be based on many more input parameters.

A. *Mechanism for Mapping of Network State with NoI*

The basic idea of using NoI for INM network state information can be summarized in the following procedural description:

1) Some state information (subtype "special condition", i.e. with relevance to other nodes, for example a failure of a hardware component) is emerging at a specific network element (NE) or link between NEs

2) The affected NE (or another one nearby that detected the failure, too) uses the NoI API (either locally on the node or at a well-known address in the network) and calls "create object", selecting a meaningful name or description of the event, for example "[link failure].[node ID].[time].[geo-position].[network name].[affected functionality]"
Note: Since the name of the event already contains most of the related content, it might in some cases not even be necessary to contact the node in order to get that information. So, in this case, NoI would indeed store this "minimal" content that can be interpreted as network "state"

The NoI machinery is now able to answer queries from arbitrary NEs, looking for matches to this specific state description, NE condition etc.

3a) A network element that is interested in the particular state of another NE would send (via the NoI upper API) either a "retrieve object" command (if it knows the object ID) already, or a "search" command to get a list of NEs that match with the state / condition that the searching NE is interested in. NoI indirection and resolution search and return all matching entries, for example within a certain geographic area or a managed part of the network

3b) Variation: Differently from (3a), the "interested" network element can passively subscribe to a certain query, and NoI would then use a "push" method to inform the interested network elements

We can distinguish active and passive retrieval mode. Both may be needed, but for different management applications. The pub-sub qualifies more for a-priori-known events and tasks, while the full search makes more sense for the unplanned.

This differentiation into active and passive might apply to the creation of objects in some sense, too, as some state can be "routinely" (e.g. periodically) distributed or published, while other state may become relevant in unplanned ways and times.

Information entities that are conforming to the NoI approach consist of a binary object and a separated locator for it (i.e. the standard case for user content). This applies to larger information entities which correspond more to the general NoI model, where NoI returns IDs of nodes that hold the requested information.

According to the note to step 2 of the procedure description, there can be a kind of minimal information entities, where the entire information is already fully contained within the identifier, so there is no need for contacting any other nodes for retrieval. It might be useful to support both types, depending on the INM use case. Alarm state could use the minimal version (saving time for retrieval via additional nodes), while support of software distribution via NoI would certainly use the full indirection model.

B. Applications for distributed storage and processing of network management information

With respect to the classical FCAPS model, we identified how distributed storage and processing for network management information can be applied to it's sub-functions. More research is necessary to analyse the individual benefits.

Fault Management

- NoI can store alarms or notifications and make them available to potentially interested nodes. It can also help to aggregate alarms in a meaningful way by pre-processing them. This would however require an NoI to allow for some active processing elements.

Configuration Management

- Maintenance: We can use NoI for a kind of "in-order-traversal" through all registered network elements or use range queries.
- It would make sense to let arbitrary nodes bootstrap themselves via NoI "bootstrap objects" which provide all necessary information while being accessible in a straightforward manner when compared to nodes that have already bootstrapped.
- Network Inventory: For every detected network element (a.k.a. "node"), a corresponding information object can be instantiated according to a specific naming / addressing convention and an ontology.
- SW updates: This corresponds probably most directly to the way end users will be using NoI when they download consumer content. In our case, the payload will be software loads (e.g. INM kernels for nodes). Using NoI for software distribution purposes for INM is quite straightforward. One issue might be the handling of different versions of the same software.

Accounting/Charging/Policies:

- Local NE state includes metering or usage data for sessions. All information needed for the generation of charging can be collected by the local network management function and get stored by means of NoI API machinery. If needed, local network management functions or other applications can get them back for the processing. Distributed storage for accounting information can allow collecting a large amount of data (statistics, session times, media, and resources usage). NoI is able to organize the information and it is possible to find it when the user must be charged.
- Today, counters play a central role in network management or operation and maintenance systems. It would be desirable if their values could be retrieved and analyzed in a more condensed form. It is possible to represent counters and their current values as NoI "counter objects".
- Policies (i.e. "trigger / action" pairs) or "rules" can be implemented as NoI objects. If they are changed, they will be directly available to all related network elements.

Performance Management:

- Key Performance Indicator (KPI) collection can benefit from distributed KPI processing which can be facilitated by treating intermediate results (e.g. from aggregation measurements) as NoI "KPI objects".

- Also KPIs can be accessed by different management functions handling different issues in the network, but requiring the same KPIs as information base.
- Time series of measured/monitored date is stored in database today, but could similarly well be stored in NoI in the future.

One of the main benefits of using NoI for INM will be to send more or less complex queries to NoI. These must be resolved according to a specific syntax, naming convention or ontology. Given that all network elements are in charge of updating the NoI objects that they created, the resolved queries will produce meaningful output, based on the indirection and resolution service of NoI.

V. Requirements and Design Considerations to Satisfy Network Management Needs

Based on the above observations we summarize the requirements for an NoI system to be useful for storing management information.

A. *Events*

NoI would require a push model for information elements such that this can be used for asynchronous events (alarms, notifications), where network management functionality needs to react on. Typically, the network management functionality of a node would subscribe to certain events that are relevant for it, so the NoI mechanism might be to subscribe to changes of information elements, or possible the creation/putting of information elements into the system, where the NoI objects do have a certain pattern/type/… The detection, processing and forwarding of events poses more or less tight timing constraints to the NoI system.

B. *Receiving ranges of information*

In general a typical network management application or function receives a range of management information to be used as input for that function like checking through the range or aggregating the information. The range can be in different dimensions up to thousands of NE, but traditionally there are two ranges of high importance.

First, the management functions are getting the same information elements from a range of managed nodes, where the range of nodes might be constrained (e.g., "Return the average bandwidth usage of each node from a certain geographic region").

Second, the range is from a time frame / time series, such as unresolved DNS requests over the last day with a granularity of 5 minutes.

A third but less important feature are conditional or qualified queries, because those can be implemented within the management functionality itself. However, an NoI with this functionality might be a more effective in gathering the information from different nodes. (e.g.: Return all the nodes having a interfaces with usage larger than 90%). The "range feature" is probably tightly linked with the internal mechanisms of the NoI machinery. It is important that the NoI mapping / resolution / retrieval functions are supporting "searches", including wild cards and multiple matches. This shows that the adoption of a simple DHT scheme based lookup is not sufficient for development of NoI.

C. *Controlled access to management information*

Depending on the persons, stakeholders, or network management functions asking for management information, results of the query might be different. This is a matter of scoping and is typically the case for different administrator levels, or internal versus external visible information. NoI is supposed to offer this controlled access, but it is unclear at the moment how that can be achieved, but the mechanisms needs to take those requirements into account.

D. *Active NoI Objects for Aggregating Queries*

Due to the hierarchical nature of the traditional NM approaches, the collection and further processing of numerous, frequent events or measurements is a significant task, also referred to as aggregation. It may be supported by NoI in the form of active NoI objects, i.e. objects that can be programmed in some way to solve network management specific sub-tasks. In case there is support for this, management information could be aggregated on the fly passing through the NoI object, or certain management functions could be implemented as active NoI objects.

E. *Security Management*

When NoI is used for storing user and account information, the security issues (such as confidentiality) required, are higher, compared to end user data; at least a strict separation must be enforced in the NoI system.

VI. Conclusion

Based on our experience with traditional network management and the novel approaches towards more decentralized network management (for example followed by the 4WARD project [1][2]), we studied the impact of also using decentralized storage systems such as the Network of Information approach, which is currently designed also in the 4WARD project. We can say that specifically decentralized management functions would benefit from also decentralized management information storage, but also the cases, where several central management applications require the same or similar information the decentralized storage can be useful.

As soon as first versions of the 4WARD NoI is developed, first tests can be made and see whether a particular NoI is able to provide enough useful storage mechanisms for storing network management information. Also a real use case should be implemented as proof of concept of such a system.

So far we also have not further detailed the issues around naming and addressing of NoI objects with management information. Or what type of management information naming might be useful in the context of NoI.

REFERENCES

[1] The FP7 4WARD Project website, http://www.4ward-project.eu

[2] Christopher Foley, Sasitharan Balasubramaniam, Eamonn Power, Miguel Ponce de Leon, Dmitri Botvich, Dominique Dudkowski, Giorgio Nunzi, and Chiara Mingardi, A Framework for In-Network Management in Heterogeneous Future Communication Networks, MACE 2008, Samos Island, Greece, September 22-26, 2008

[3] C. Dannewitz, K. Pentikousis, R. Rembarz, É. Renault, O. Strandberg, J. Ubillos, "Scenarios and Research Issues for a NoI", MobiMedia 2008, Oulu, Finland, 7-9th of July 2008

[4] J. Schoenwaelder, "Protocol-Independent Data Modeling: Lessons Learned from the SMIng Project", IEEE Communication Magazin, vol. 46, no. 5, May 2008.

[5] J. Strassner, Directory Enabled Networks. Indianapolis: Macmillan Technical Publishing, 1999

[6] C. Adam and R. Stadler, "Patterns for Routing and Self-Stabilization", 9th IEEE/IFIP NOMS, Seoul, Korea, Apr. 2004.

[7] Y. Yemini, G. Goldszmidt, and S. Yemini "Network Management by Delegation", Proceedings, IM'1991.

[8] D. Levi, J. Schoenwaelder, Definitions of Managed Objects for the Delegation of Management Scripts, RFC 3165.

[9] Carroll, R., Fahy, C., Lehtihet, E., van der Meer, S., Georgalas, N., & Cleary, D. 2006, "Applying the P2P paradigm to management of large-scale distributed networks using a Model Driven Approach", in Proceeding of the 10th IEEE/IFIP Network Operations and Management Symposium (NOMS 2006), Vancouver, Canada.