Copyright © 2010 IEEE.

Citation for the published paper:

Title: Experimenting with Infrastructures

Authors: Björn Ståhl, Raphael Caire, Luc Le Thany, Rune Gustavsson

Conference: The Fifth International CRIS conference on Critical Infrastructures – Interacting Critical Infrastructures for the 21st Century, Beijing, 2010

This material is posted here with the permission of IEEE. Such permission of the IEEE does not in any way imply IEEE endorsement of any of INTEGRAL partners' products or services Internal or personal use of this material is permitted. However, permission to reprint/public this material for advertising or promotional purposes or for creating new collective works for resale or redistribution must be obtained from the IEEE by sending a blank email message to pubs-permissions@ieee.org

By choosing to view this document, you agree to all provisions of the copyright laws protecting it.

Experimenting with Infrastructures

Björn Stahl* Luong Le Thanh† Raphaël Caire‡ Rune Gustavsson§

***Abstract*—Laboratory environments for experimenting on infrastructures are often challenging both technically, politically and economically. The situation is further complicated when the interaction *between* infrastructures is in focus rather than the behaviours of a singular one. Since ICT often has a key role in experiment management, data gathering and experiment upkeep – controlled experimentation becomes even more dif cult when some of the interactions studied are between ICT and another infrastructure. The work described herein concerns design, implementation and lessons learned from the construction of a joint-effort experiment environment for, essentially, *experimenting with infrastructures*.**

The layout of this article is as follows; In *Background* we provide problem descriptions, background and history relevant to this work, combined with a brief summary of previous efforts. Thereafter, *Experimenting with Power grids* details the design of the power grid laboratory environment and some of the challenges involved when experimenting within that particular domain. This is followed by a corresponding section, *Experimenting with ICT* which describes our general approach to creating robust software environments supporting controlled experimentations. The main section then, *Experimenting with Power grids and ICT* combines these two environments into a unified experiment environment wherein we can study the interfacing between different critical infrastructures. In *challenges* we briefly describe some issues, both open and closed, that appeared while establishing this environment. Lastly, in *opportunities* we elaborate on some of the possibilities that this particular setup provides.

I. Background

The need for strong experimentation, verification and validation efforts able to transcend traditional infrastructural, and scientific, borders is a large one – if we ever are to successfully restructure and improve current critical infrastructures to fit and surpass current and upcoming challenges that is. To this end, the settings that enable such efforts will also need to forego a similar enchantment, both in a macroscopical and a microscopical sense. With a practical focus in this regard, we herein look at the *interfacing between infrastructures* at two similar, but distinct, levels; One being the interfacing between power grids and ICT, and the other level being the interfacing between experiment environments for said infrastructures. The underlying motivation is partly fueled by the european FP-6 project INTEGRAL[1][1] and the SEESGEN-ICT[2] thematic network[2]. The INTEGRAL case actually covers the integration and interfacing of three different critical infrastructures, i.e. the electric grid including renewables, a customer-oriented business process infrastructure and a SCADA (Fig. 1) - ICT one. The SCADA - ICT case is here of particular interest as we in part showcase ways of improving *resilience* through the implementation of *self-healing mechanisms* as a response to some harmful or even catastrophic event. This is especially relevant because of the inherent coupling between the SCADA and the grid – and that any event affecting the grid in such a way that some form of remedial action is needed, also may adversely affect the ICT support needed to perform such actions.

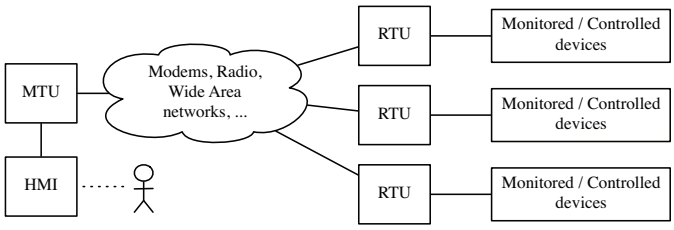

Fig. 1. A basic Supervisory Control and Data Acquisition (SCADA) model, wherein a series of Remote Terminal Units (RTUs) acting as telecommand/telemetry devices are connected to the governed infrastructure. The RTUs are in turn connected to a Main Terminal Unit (MTU) using a variety of communication technologies and protocols. The interaction with the MTU is controlled using some sort of Human- Machine Interface (HMI) which provides operators with a view into the dynamic state of the infrastructure.

While-as we cover both the design of an experiment environment for a microgrid - ICT and a different one for an ICT - ICT setting, there is quite some bias placed towards the ICT - ICT problem in particular, partly as in this context, ICT and its actual role is the least understood one but also because of the recursion involved; the need for ICT as means for observing and intervening with ICT. In terms of related work or other approaches, the environment presented in Sect. III has a similar idea and goals in terms of overall architecture as the NSF GENI[5] efforts (Fig. 2). A contrasting frame of reference between the work herein and in GENI can thus be found through planetlab[13][6] but mainly concerns scale and application domain specificity.

II. Experimenting with power grids

This section describes the setup behind an experimental environment for, amongst other things, self-healing microgrids as per the description in the previous section and will serve

*Blekinge Institute of Technology, Karlskrona Sweden. E-mail: bjorn.stahl@bth.se
†Grenoble Institute of Technology, Grenoble, France. E-mail: luong.le-thanh@g2elab.grenoble-inp.fr
‡Grenoble Institute of Technology, Grenoble, France. E-mail: raphael.caire@g2elab.grenoble-inp.fr
§Blekinge Institute of Technology, Karlskrona, Sweden and the Royal Institute of Technology, Stockholm, Sweden. E-mail: rune.gustavsson@bth.se

[1]Integrated ICT-platform based distributed control in electricity grids
[2]Supporting Energy Efficiency in Smart GENeration grids through ICT

978-1-4244-8081-4/10/$26.00 ©2010 IEEE

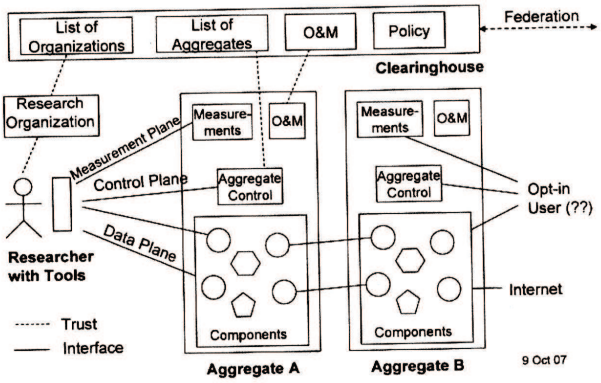

Fig. 2. NSF GENI Architecture

as a baseline and fundamental environment that the other sections of this paper will expand upon. It covers three distinct parts; The physical distribution network, the agent logic and sensoring equipment enabling self-healing, and finally the SCADA system itself.

Fig. 3. A model of the microgrid

The analogical micro distribution network, illustrated by Fig. 3, was sized by aggregating some electrical nodes of a real distribution network having 30 MVA rated power at 20 kV. In order to best represent the behaviour of a real network for many RTUs, and satisfy economical conditions, the network of test bench of 30kVA, 0.4kV was adopted. The scale reduction of the microgrid components was carried out through different ratios (power, inertia and voltage) to assure the similarity of the internal static and dynamic behaviours of the network which includes aggregated loads, network components (on-load tap changers for instance) and DER/RES. A strong requirement is that the performance of the control systems must be unchanged in comparison with the real system. This network has 14 nodes, 17 lines, 10 loads and 6 RER/DERs, divided into several areas which represent different network characteristics. On top of this network, there is considerable supporting ICT infrastructure illustrated in part by Fig. 4 and in part by Fig. 5.

A fault location algorithm had previously been developed using MATLAB, and subsequently, the governing agent was also built using a combination of MATLAB and MATLAB OPC toolbox. The OPC Data Access standard over Ethernet

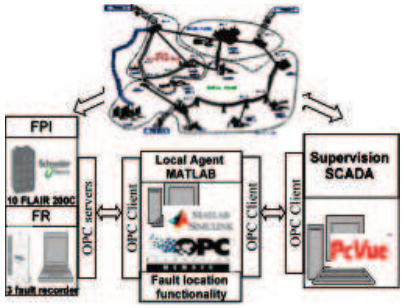

Fig. 4. ICT dependent structures overlayed on the microgrid model

provides for the common protocol for communication between the necessary OPC server associated with RTUs (which uses TCP/Modbus) and between the SCADA control center and the OPC client (MATLAB OPC Toolbox).

Communicating RTUs such as the Fault Recorder emulators (developed within LABVIEW) as well as the Fault Indicators (Flair 200C[3]) are directly connected to PCs that then are further interconnected across several local-area networks – with the application level communication being supplied by the OPC client/server solution. Finally, the advanced control and batch execution used to accomplish the self-healing functionalities can be carried out either directly by the local agents *or* indirectly from commands issued by a distributed service operator, DSO, although empowered from the information supplied by the agents.

The composition of the OPC real-time communication system for the INTEGRAL demonstrator is shown in Fig. 4 and the computation that regulates the system comes from the OPC Client Toolbox and the MATLAB local agent with its embedded fault location algorithm. In terms of data acquisition, there are numerous current and voltage sensors implemented. The dynamic data on distribution network behavior gathered from these sensors are continuously exchanged with the IEDs (Intelligent Electronic Devices); here limited to just the Flair 200Cs and Labview Fault Recorders. These are dedicated to the remote monitoring of middle to low voltage substations. When a passage of a fault is detected, the current and voltage data are recorded in real time and transmitted to Agent MATLAB via the OPC server[4]. The data exchange is then performed between the OPC server and MATLAB OPC Client toolbox. Afterwards, the exact faulty segment will be determined by the fault location algorithm. The computed action, i.e. which circuit breakers or switches that are to be opened or closed in order to restore as much load as quickly as possible after the fault have been identified, is then relayed from MATLAB to the protection and automated devices in the distribution network via the SCADA system.

A. *Communication between IEDs and SCADA software*

: In order to supervise, control and communicate with each and every automatic device and software, the supervisory

[3]These are fault passage indicators furnished by Schneider Electric Telemecanique

[4]For which a Schneider Electric OPC server product called OFS (OPC Factory Server) is used.

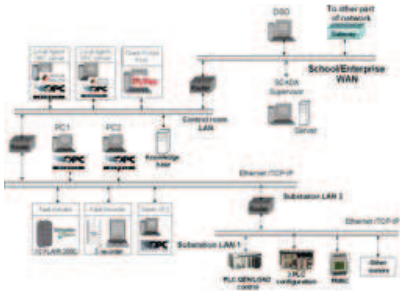

Fig. 5. early overview of the ICT solution

software PCVue has been adopted[5] and allows for the support of a very wide range of industrial SCADA protocols. The communication between the supervisor and the process equipment such as PLCs (programmable logic controllers) is handled by a component called the *Supervisor Communication Manager*. The communication technology involved includes; OPC, native driver equipment protocol, DDE, lanworks, ... In the self-healing demonstration, the OPC technology is used to communicate between supervisor and local intelligent agent, and the native driver equipment protocol is used to link the industrial PLCs.

The equipment devices are labeled *nodes* of a particular network. The messages that pass between the Supervisor and the process equipment are called *frames*. The mechanisms and boundaries which allows the supervisor to interact with the MATLAB Intelligent Agent and the industrial PLCs are as follows;

- 16 simultaneous communication channels, each with its own protocol.
- The refresh-rate is fixed to the rate specified by the frame-scan rate.
- Real-time kernel between the supervisor and the process equipment – periodically refreshes the values of variables in the Supervisor database using data from communication frames.
- Each entry in the Supervisor database which corresponds to an equipment source is further linked to a specific location in the communication frames. At least one corresponding entry has to exist for a link to be established between a frame and the database.

III. Experimenting with ICT

There are several deeply rooted issues when experimenting with ICT and many of them stem from the extremely heterogeneous and dynamic nature inherent to software-intensive systems. This section will briefly discuss some of the generic, overarching problems that need to be taken into account when seriously experimenting with- and using- software. We will thereafter describe the overall concepts of a system for generating and maintaining software- based experiment environments to strengthen control and improve accuracy of gathered data.

[5]PcVue is a SCADA solution for multi-station supervision and control, developed based on the considerable industrial automation sector of the ARC Informatique Company, and as such is representative of current state-of-the art in SCADA

Modern software is strongly structured around various hierarchical form of separations that are tied to some abstraction or to the semantics of surrounding systems. Some of these are enforced by the technology that makes software run, *execute*, as part of performing computations. Others are simply modeled in ordered to, in some way, assist the development of software; there is however, a certain degree of overlap between the two. For sake of reference, the abstractions that are strictly modeled are herein called *static* and can as such be studied and processed by tools and methods other than computers and many such separations are simply stripped away or reconstructed, optimized, into something more efficient during the translation from human-readable formats, source-code, into a format efficient for computer execution, binary-code. Furthermore, the effect these static abstractions may have on program behaviour can, to a fair degree, be predicted[6] and determined benign or undesired in advance – using formal techniques such as model checking and computer proofs, or for that matter, be projected over an modeled machine and simulated. An often held fallacy in this regard however, is that the source code used to construct a software system is strongly representative of the program(s) that will execute on a computer[8]. For the other category, *dynamic abstractions and separations*, observable computation patterns, i.e. behaviours, are by and large undeterminable up until essentially the point where its corresponding execution is performed because executing software exhibit advanced characteristics such as *non-locality*, *heterogeneity*, *recursiveness*, *polymorphism*, *re-connective* and *concurrency* in addition to an very large space of possible states – all of which are influenced by the information the system receives, processes and transmits. This is further complicated by low-level optimizations in processors and similar components using caching and prefetching of code and data.

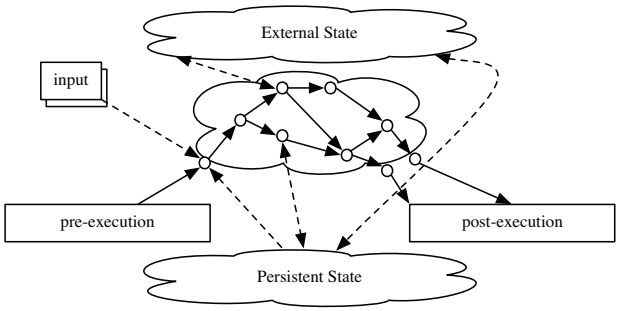

Fig. 6. The principal problem of virtualization.

To partition this challenge into reasonably sized chunks, strategies are employed both statically and dynamically to enforce the separation of logically disjoint such chunks. The dynamically enforced separations are referred to as *virtualizations* and can be found at a variety of levels of granularity wherein the more commonly known one is the notion of *processes* in modern operating systems. The separation pro-

[6]Albeit the limitations that stem from the age-old computability 'halting problem' still apply.

vided even in those cases are however, to a varying degree, insufficient measures to safe-guard against the aforementioned execution characteristics. As shown by Fig. 6, ideally the full execution of a bounded computation, virtualization, is dictated solely by its initial configuration (its code and its input), but there are dynamic sources of information that necessarily are external to the virtualized space[7] but influences the execution in such a way that the protective enclosure can be breached and means of intentionally doing this is a currently active area of research in software security[8]. Thus, even though the intended target for experimentation is encapsulated using some form of virtualization, an important step in initial experimentation is to isolate and control such state-holders.

Means for performing controlled experimentation on fairly strong virtualizations, such as processes, which contain single, confined programs are well developed and numerously and to a lesser extent larger borders such as a complete machine, but when the form is less traditional – as the case with software-intensive systems – the tools are far less advanced.

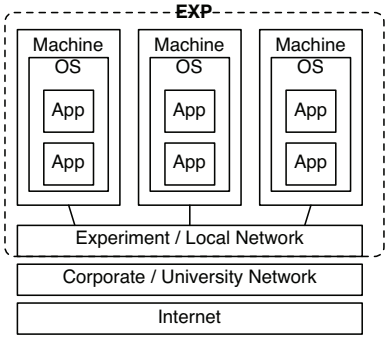

Fig. 7. EXP-I and its respective borders.

As an approach to expanding this control to incrementally larger borders, a solution called `EXP`[12], was developed as part of previous projects, like described in the introductory section. The Borders of `EXP` is illustrated by Fig. 7. Briefly put, `EXP` is partly a hierarchical data-model describing abstract roles that can then be assigned to lab-nodes in the environment, and partly a set of services which combined enforces the policies defined by the roles unto the nodes. The major services, as shown in Fig. 8 are thus;

- *Startup* - Role-specific bootloader sent over the network to affected nodes to control node startup, used both to run integrity / hardware checks, as an enabler for other services and to activate the current configuration in a controlled manner.
- *Restoration* - Provides the ability to generate snapshots of the inactive state of a node but also the ability to revert to previous snapshots.
- *Experimentation* - Miniature, low-footprint FreeBSD based- OS for quickly deploying small agents to act as input or noise (network traffic generators and the likes) to main nodes.

[7]They don't even strictly have to be a part of the software-system at large, but may as well come from the surrounding information system or a user

[8]This is a rather brief and shallow summary of the problem. The full extent of this discussion is however, well outside the scope of this paper.

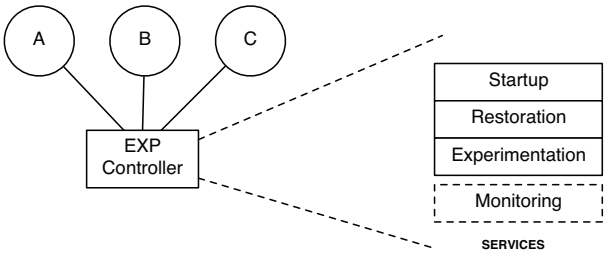

Fig. 8. EXP-I services, monitoring added in EXP-II.

In addition to the nodes used for experiments, there is also an additional node reserved for coordinating the others. This node is called a *controller*. The responsibilities of providing and managing services and nodes are primarily put on the controller. This enables two distinct modes of operation; *Deployed environment* versus *Sustained environment*. In a deployed environment, the nodes have physically been hooked up to the controller for configuration. When configuration has been completed for all involved nodes, the controller is either disconnected from the network or reverted into acting merely as a network router. By contrast, in a sustained environment, the controller actively micromanages the nodes[9] involved as well as acting as intermediate storage.

Expanding on these ideas, the `ICT` part of the environment detailed herein takes two geographically separated, sustained `EXP` environments and combines them into one larger environment called simply `EXP-II` that can alternate between a deployed and sustained setting as well as introduce monitoring services using run-time tracing technology such as DTrace[9] combined with post-mortem analysis for both online and offline data-acquisition on ongoing or completed experiments.

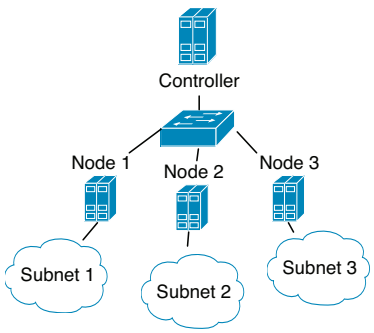

Fig. 9. The basic lab setup.

The initial structure for the `EXP-II` environment is depicted in Fig. 9 and corresponds to an `EXP` controller, an IEEE 802.1Q (VLAN) capable switch and three lab nodes that each will manage a subnet, corresponding to what was shown in Fig. 4 and Fig. 5 respectively. The quality of the nodes and their parts were on the level of common, off-the shelf components (COTS). Two such setups were created, one for each of the two geographical locations. This model will be expanded upon in the next section.

[9]In some cases this includes their power using programmable outlets

IV. Experimenting with power grids and ICT

Taking advantage of both experiment environments as detailed in the previous sections, the task then becomes to combine the two into a unified experiment environment – this while still maintaining the respective benefits, reductions and structure of the individual environments but at the same time open up for new opportunities without compromising functionality, integrity or security.

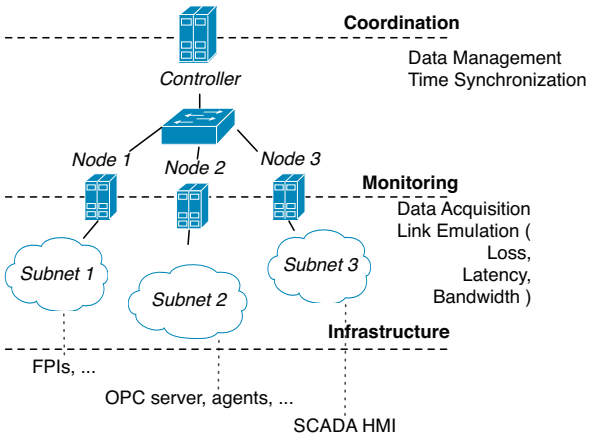

Fig. 10. Initial step, experiment environment

A. Isolated operation

To this end, an incremental approach was ultimately chosen. The first step, was to define and construct a software configuration which would virtualize the communication between three larger slices of the SCADA system in the microgrid. To this end, the abstract experiment environment detailed in Fig. 9 was instantiated as shown in Fig. 10. Briefly put, the configured roles for the three nodes were set to be running FreeBSD**??**, acting as *bridges* with the option of having dummynet[10] configured for both directions across the bridge, in order to be able to emulate a variety of types of communication links and faults. A major criterion for this stage was for the environment to operate as an isolated cell, meaning that there would be no access to external communication through a corporate *WAN* on the internet. When the environment is in an active state, meaning that the experiment or demonstration is running, the controller acts as part router between the three nodes, using static routes, but also as part aggregator for acquired data. The actual data acquisition however, is gathered as raw packet recordings on each individual nodes on the interface connected to the monitored subnet rather than in the confines of the `EXP-II` network as such.

B. Joint operation

The second step then, was to establish the environment in a *joint*-operation mode. In this mode, we have a situation with two controllers being connected to an external, likely hostile, environment; making security concerns even more pressing. To deal with this situation, and to establish a trusted connection, the respective firewalls *by default* refuses all incoming traffic. Then at agreed upon points in time, the remote location opens up for a single incoming connection from a fixed IP address. The other location initiates the connection through which a VPN tunnel is established, using pre-shared keys, that have been exchanged offline. The difference in terms of information flow, is now that the acquired traffic is being forwarded to the remote location rather than being stored locally. Dummynets can still be used, but the default setting here is having them disabled.

C. Unified operation

The third step was to establish the environment in a *unified*-operation mode, as shown by Fig. 11 building upon the joint-operation mode but with several major changes. For starters, the bridging nodes at the local site no longer perform any data-acquisition or traffic shaping (dummynets). Secondly, when the VPN tunnel gets established, the static routes that previously joined the three nodes together at the local site, are altered in order to redirect traffic to take a 'detour' through their respective analogs in the remote lab. For instance; traffic going from a FPI through (site a, node 1) destined for the OPC network, will follow the path;

`FPI`→ (site a, node 1) →(site a, controller) → (site b, controller) → (site b, node 2) → (site b, controller) → (site a, controller) → (site a, node 2).

The return-path follows the same general pattern.

In closing, the overall principle behind these three modes of operation mimics some of the ideas powering the microgrids as well; during ideal conditions (unified operation) information is behind traded between cells (here represented by the two physically separate environments) in a seemingly coupled fashion. Should some event disturb or threaten this setting, the system reverts to a safer mode of operation (joint operation) and should things deteriorate even further, they can be switched to isolated operation. While not currently taken advantage of, this could be an interesting avenue to explore further down the line.

V. Challenges

The purpose of this section is to hi-light partly the less obvious challenges involved in constructing the environment depicted by Sect. IV but also to discuss issues that for practical reasons where left open or that may prove relevant to projects facing similar problems.

A. Technical Barriers

Constructing the unified environment proved challenging in several respects. When the separate problems of experimenting with power grids and experimenting with ICT are accounted for, the obvious and challenging part is the inherent coupling between the power grid and its managerial ICT and the herein assumed brittleness of the devices and protocols involved in the SCADA process. As the risk for harming equipment through mis- or mal- configured ICT have previously been

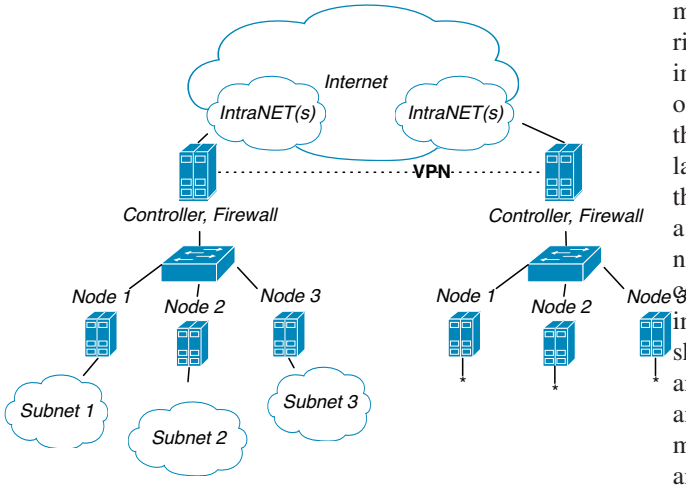

Fig. 11. Unified Operation Overview

shown to be uncomfortable at best[3] coupled with the volatile nature of software behaviour[10], there are technical barriers which – until cleared through validation, implementing or hardening of safe-guards on border conditions – puts heavy restrictions not only on experiments themselves, but on software and network security as well.

Furthermore, the implicit maintenance requirements regarding the equipment that comprise the physical layer(s) of the experiment environment brings forth additional complexities ascertaining that the two environments have a synchronized configuration to as fine a granularity as possible. One such issue that appeared concerned a race condition through the use of a KVM to switch between active nodes when working on several components in an intermittent fashion. As per the principal problem of software experimentation in a para-virtualized setting illustrated by Fig. 6 the activities of a KVM is merely one such external state-holder that cannot easily be precisely controlled or manipulated, thus have to be both cautiously treated and manually verify that end-results were not causally linked to the state or function of such state holders.

B. Economical Barriers

The ICT experiment coordination facility as depicted in Sect. III only has managerial control over the three routing nodes in the main lab along with the corresponding devices in the analogous lab, i.e. it cannot not directly roll-back or in other wise control[11] the components outside of this abstract perimeter. While the perimeter by all means can be expanded using more specialised technology and software, this has not yet happened and it is not an immediate goal. For this to happen, we speculate that the reasonable, incremental, enhancement would first be to improve the granularity of monitoring to cover not only communication against fairly rigid interfaces, but also the comparably dynamic and adaptive interactions inside the SCADA of the grid-level ICT. Another strong economical barrier with technical undertones is the activity of maintaining two instances of the same ICT laboratory environment, for two main reasons; One being that domain expertise may be geographically tied to one (in a collaborative project such as the one depicted herein) or none of the geographical instances (outsourced) which implies certain overhead in response times in regards to troubleshooting and maintenance. The second concerns the comparably short longevity of core components such as hard-drives[14] and memory capsules[15] which are susceptible to the most amount of stress. While cheap and easily replaceable, such maintenance again needs to be synchronized between the sites and considerations taken in regards to possibly influential entropy (such as wear-leveling in flash-based storage devices), if even detected[7].

C. Political Barriers

Additionally, due in part to the incremental development of the distinct (Sect. III, Sect. II) environments, there is an additional dependence in the former case on the intermediate ICT infrastructure of the respective local area network environments at both ends. Although virtually separated through technologies such as VLAN tagging, the environments both rest on preexisting networking infrastructure, including LAN/WAN border management such as firewalls and intrusion detection systems. When, as in our case, the larger WAN is the Internet, border management policies tend to be very strict and it is therefore likely that incompatibilities arise between such policies when the networks and policies have been developed independent of each other. Ultimately, this can be construed as a political barrier that has strong technical consequences. For the particular scenario described herein, the major limitation was the need to essentially tunnel TCP over TCP rather than TCP over UDP for the VPN, which implies undesired interactions when the TCPs retransmission algorithms are being applied recursively; a problem likely to result in a noticeable decrease in network performance. The extent of this problem is lessened however, by the distinction between *isolated operation* and *unified operation* since the different modes complement each other.

VI. Opportunities

In addition to the benefits than can be reaped in respective domains from the individual infrastructure environments, there are quite a few interesting opportunities that open up when combining the structure from Sect. IV with the considerations in place from Sect. V. This section will be used to elaborate on a few of these that will be of interest in the near future.

A. Protocol Design and Evaluation

There are lots of protocols involved in current SCADA systems, ranging from small or narrow and product specific to broad and generic. Given the fairly radical suggestions as to

[10]More specifically, as far from all processing circuitry have clear defined and tested reactions to arguments received as part of communication protocols

[11]Albeit that through intervening with their communication via the virtualization borders also to some extent intervene with the system, but the control is coarse-grained

the future of the grid(s), there are justified concerns regarding how some of these protocols would behave in new settings, which restrictions they impose on supporting communication and processing technologies, exactly which information is communicated and similar aspects. The data-sets that will be obtained from *isolated operation* can be used to specialize or generalize available protocols to fit new grid structures such as the microgrid, but also as input to the design of future network protocols and to the configuration of security equipment such as firewalls and intrusion detection systems.

B. ICT Monitoring Models

The complex interactions between information processing systems, information systems and the physical grid have been a source of much commotion, and one do not need to look further than the northeastern blackout of 2003 for strong examples to that effect, and the work involved in discovering underlying causes behind such events are further complicated by complexity of software analysis and debugging. With more adaptive ICT systems, more modern network structures and dynamic services models there will be an even stronger incentive to monitor not only these different layers individually as is currently done, but to be able to monitor them in relation to each other. The setup that has been covered will serve as ample grounds for the development of such monitoring models, technologies and methods that essentially allows the SCADA concept to be applied recursively, i.e. SCADA for SCADA ...

C. Rogue SCADA

That SCADA systems have a dodgy past in terms of security is a well-known affair, and it seems unlikely that we somehow will be able to retrofit major developments in software and information security to be usable in current closed and legacy-rich SCADA systems. Provided the transition from a supposedly closed network to being transported over comparably more open channels, even though the information itself may be encrypted, opens up both new attack vectors and allow for more fine-grained attacks, i.e. a specific cell can be targeted in a denial-of-service manner rather than compromising an entire infrastructure. The structure used herein, allow for evaluating the consequences of various kinds of directed attacks and their corresponding protections both from the perspective of an antagonist outside the system e.g. what can be discerned from side-channels in the data-flow, but also, when using the unified mode, what the effects of a compromised cell could be.

VII. Conclusions

In conclusion; we have introduced tools and means for the controlled experimentation on the interface between ICT and energy transmission in a critical, self-healing context – which we can use not only to properly evaluate means for improving resilience, but also obtain much needed datasets on systemic behavior in several situations characteristic for future smart- and micro- grids. To this end, we have also identified several barriers relevant for those working on similar targets. While the majority of the work described herein are completed or very near completion, work for the proximate future involves exploring the opportunities mentioned, but also to attempt to generalise this solution to fit other experiment-oriented endeavours and needs from similar domains.

Acknowledgements

This work is partially sponsored by the European Union EC grant F66-038576 "INTEGRAL". The authors would further like to acknowledge the following people for their assistance in the development of these environments; Per Mellstrand, Johan Zackrisson, Charlie Svahnberg and Shahid Hussein.

References

[1] [Online]. Available: http://www.integral-eu.com
[2] [Online]. Available: http://seesgen-ict.erse-web.it/
[3] [Online]. Available: http://www.cnn.com/2007/US/09/26/power.at.risk/index.html
[4] "Freebsd." [Online]. Available: http://www.freebsd.org
[5] "Geni - global environment for network innovations." [Online]. Available: http://www.geni.net
[6] "Understanding and resolving conflicts on planetlab," 2008. [Online]. Available: http://www.cs.princeton.edu/ llp/policy.pdf
[7] L. N. Bairavasundaram, G. R. Goodson, B. Schroeder, A. C. Arpaci-Dusseau, and R. H. Arpaci-Dusseau, "An Analysis of Data Corruption in the Storage Stack," in *Proceedings of the 6th USENIX Conference on File and Storage Technologies (FAST 08)*, San Jose, California, February 2008.
[8] G. Balakrishnan, T. W. Reps, D. Melski, and T. Teitelbaum, "Wysinwyx: What you see is not what you execute," in *VSTTE*, 2005, pp. 202–213.
[9] B. Cantrill, M. W. Shapiro, and A. H. Leventhal, "Dynamic instrumentation of production systems," in *USENIX Annual Technical Conference, General Track*, 2004, pp. 15–28.
[10] M. Carbone and L. Rizzo, "Dummynet revisited," *Computer Communication Review*, vol. 40, no. 2, pp. 12–20, 2010.
[11] R. Gustavsson and B. Stahl, "Self-healing and resilient critical infrastructures," in *CRITIS*, 2008, pp. 84–94.
[12] P. Mellstrand and R. Gustavsson, "Experiment based validation of ciip," in *CRITIS*, 2006, pp. 15–29.
[13] L. L. Peterson, A. C. Bavier, M. E. Fiuczynski, and S. Muir, "Experiences building planetlab," in *OSDI*, 2006, pp. 351–366.
[14] B. Schroeder and G. A. Gibson, "Disk failures in the real world: what does an mttf of 1,000,000 hours mean to you?" in *FAST 07: Proceedings of the 5th USENIX conference on File and Storage Technologies*. Berkeley, CA, USA: USENIX Association, 2007, p. 1.
[15] B. Schroeder, E. Pinheiro, and W.-D. Weber, "Dram errors in the wild: a large-scale field study," in *SIGMETRICS 09: Proceedings of the eleventh international joint conference on Measurement and modeling of computer systems*. New York, NY, USA: ACM, 2009, pp. 193–204.