Network Resource Control for Grid Workflow Management Systems

[1,2,4]Rudolf Strijkers, [1]Mihai Cristea, [1,5]Vladimir Korkhov, [3]Damien Marchal, [1]Adam Belloum,
[1]Cees de Laat, [1,2]Robert Meijer,
[1]Universiteit van Amsterdam, Amsterdam, The Netherlands,
[2]TNO Informatie- en Communicatietechnologie, Groningen, The Netherlands,
[3]CNRS: Centre National de la Recherche Scientifique, Lille, France,
[4]ETH Zürich, Zürich, Switzerland,
[5]St. Petersburg State University, Russia

Abstract – **Grid workflow management systems automate the orchestration of scientific applications with large computational and data processing needs, but lack control over network resources. Consequently, the management system cannot prevent multiple communication intensive applications to compete for network resources, which leads to unpredictable performance. Currently, the lack of control over network resources may prevent certain applications, i.e. applications that need high capacity and Quality of Service, to utilize Grids. Hence, such applications would use dedicated infrastructures. Because the costs to build dedicated infrastructures may far exceed the cost of using existing Grids, the Grid needs to support mechanisms to optimize the interworking between networks and applications. In this paper, we present the architecture and proof of concept to control network resources from Grid workflow management system and to manage network resources from workflow-enabled applications at run-time. Depending on the current network infrastructure capabilities or future advances, applications may employ existing QoS mechanisms or use application-specific ones to provide the desired network service. We believe that our approach leads to performance improvements in communication intensive applications and enables novel Grid applications, which require optimal interworking between networks and applications.**

Keywords: Distributed Computing, Network Management, Workflows

I. Introduction

Grid workflow management systems enable smart utilization of computational resources in Grid environments by allowing scientists to plan, schedule and run complex application execution scenarios as part of their scientific experiments. Resource virtualization, i.e. resource as a service, is one of the basic design principles in Grid architecture to make the large amounts of resources manageable and easier to use by scientists. Because most Grid applications have large computational demands, the attention in Grid computing has predominantly been focused on effective and efficient sharing of computational resources.

In recent years, many initiatives have emerged, in which researchers collect enormous amounts of data from the environment, such as dikes [1], the sky [2] or from scientific experiments, such as CERN's LHC detector [3]. By using the Grid, a large amount of resources are at the disposal for such applications, which would otherwise be technically or financially unfeasible to achieve with dedicated systems. The term Sensor Grids [4] loosely defines these types of applications.

Sensor Grid applications are difficult to realize from the network perspective, because they can only execute well, if the underlying network supports their communication demands. On one hand, applications such as e-VLBI, only need high-speed network connections at the time of an experiment, but the required link connectivity may change while the experiment progresses. On the other hand, an early warning system for dike failure might need to redirect sensor data to intermediate nodes in the network for filtering or aggregation before feeding it to computation nodes. Because sensors cannot know in advance to where the data needs to be sent, the network has to be configured on beforehand or adapted at run-time. Unfortunately, such behaviour is hard to achieve in current Grids, because networks do not expose their resources and services to the application domain.

Here, we present the architecture and a proof of concept to control and manage existing network services, such as MPLS [5] or deploy application-specific ones from Grid workflow management systems. The novel idea of our approach is that network elements are virtualized as software objects in the application domain. The application programmer uses the programming interface of the software objects to implement a desired network service. At run-time the workflow management system deploys the application-specific network service on the network elements, which enables applications to control the network elements. Grid workflow management systems are a natural choice to implement these mechanisms, because they provide abstractions to consider networks as a collection of software objects and already provide similar functions to manage computational and storage resources. Therefore, it is straightforward to reuse and extend existing workflow management system with control over networks. To our knowledge, no architecture or framework is described in literature to control network resources from workflow management systems.

The paper is organized as follows. In Section II, we introduce the problem domain, design issues and the basic framework. In Section III, we present a proof of concept using

WS-VLAM [6] workflow management system and in Section IV we present preliminary experiments and results to demonstrate the feasibility of network control from workflow management systems. Related work is presented in Section V, followed by discussion and future work in section VI. The paper concludes with Section VII.

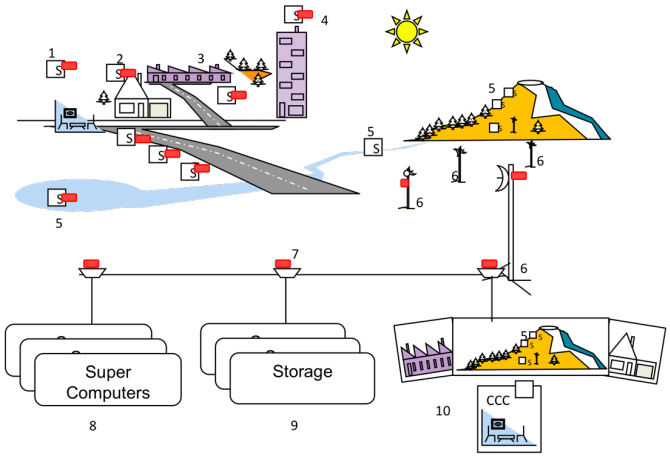

Figure 1. Three major components form a large-scale observation system: sensors (1-5), computers (8-10) and interconnection networks (6, 7).

II. Interworking Between Sensors, Networks and Grids

Typically a Sensor Grid application is composed of three main components: (1) sensor networks that monitor environmental properties, (2) computers that process sensor data and (3) an interconnection infrastructure that connects sensors to computational resources, either by using the Internet or dedicated networks.

Depending on type of the observation (Figure 1), sensors (S) need to be distributed at specific locations (e.g. homes (2), cities (2, 3, 4) or in rural areas (5)). Depending on their situation, sensors are connected via: sensor-to-sensor network, wireless (6) or wired (7) dedicated network or the Internet. When connected to the Internet, applications only get best-effort connectivity. In contrast, dedicated networks can support software plug-ins for application-specific data transformations, such as aggregation, filtering or pre-processing of data streams or conversion of sensor network protocols into Internet or application-specific protocols. Such networks, or the Internet where necessary, connect sensors to supercomputers (8), storage (9) or management systems (10).

Workflow management systems (WMS) provide a transparent and flexible way to compose and execute distributed applications on the Grid. An intuitive approach would be to use a WMS to orchestrate all the components of a Sensor Grid application, i.e. let the WMS care about execution of software components on distributed computational resources and the management of interdependencies and data transfers between these components. But Sensor Grid applications have specific demands that have to be addressed by WMS before an implementation is feasible. The most Sensor Grid demand is to reserve and control network resources to ensure that sufficient network capacity is available to transfer data to computational nodes. WMS already fulfil the task for computational and storage resources. What are the requirements to include network control?

A. Combined Allocation of Network and Grid Resources

The applications we are considering exhibit a strong sensitivity to their execution time; they are all connected with sensors, record and process real-life data at a given sampling rate. When a sensor produces data, it either has to be stored or processed immediately. Once a radio telescope turns on, for example, it is necessary to receive and process the data as it was transmitted. When this is not fulfilled for a moment, important events may be missed, wrong correlations may be made, and the whole experiment may fail. We consider such applications as *time-critical*. Time-critical applications need to have insurance that the environment is properly dimensioned at run-time. Because Grids can support reservation of resources on beforehand, Grids can provide this insurance. However, in order to support experiments that involve sensor networks, the interconnection networks, shared or dedicated, need to support resource reservation and control too.

In the case where resources are not known on beforehand, for example when an application needs to switch to another

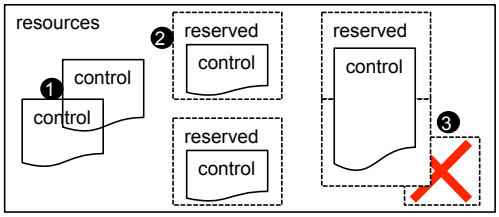

Figure 2. Without reservation network services can conflict (1). Reservation guarantees resources to be available at run-time (2). In order to support dynamic reservation, a resource manager needs to be able to override other reservations when needed (3).

data source at run-time, the resource manager has to provide a function to potentially override the reservations made by other applications, i.e. the system has to support rank ordering of the applications in order to decide which applications have priority above others (Figure 2). In addition, the network and computing resources need to be reconfigured on the fly to adapt to the new situation. Because only the application programmer knows how to organize the resources for the application, it needs mechanisms to manage computational resources as well as network resources. Therefore, the resource manager should be accessible through an application programming interface.

In the Grid the exact locations of the Grid nodes are unknown at reservation time. Therefore, we need to match and reserve network resources after the Grid nodes are allocated. If the network cannot provide the required resources for the application, the computation nodes need to be rescheduled. This process involves coordinated interaction with Grid brokers and network managers and can be dealt with from a WMS. An additional benefit of using a WMS is that it can negotiate and determine the best Grid brokers to submit jobs to, based on statistics gathered from previous submissions. In

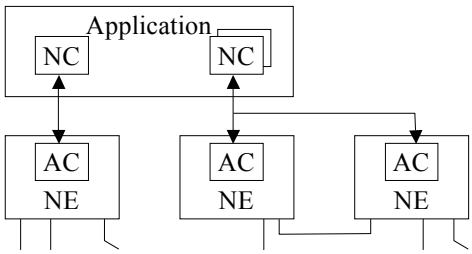

Figure 3. The components of a general programmable network architecture.

addition, WMS specialize in dynamic and advanced algorithms for matching, reserving and managing resources.

B. The network as software object in Grid Applications

Networks need to provide enough flexibility to support network reservation and management required by our class of Grid applications. In [7] we presented a concept in which network elements are virtualized as software objects in the application domain. Our concept regards individual Network Elements (NEs) as resources, which are exploited directly or through the Internet as software components in application programs. A NE component (NC) can be seen as a manifestation of the NE in the application, i.e. a virtualized NE. Consequently, all virtualized NEs together create a virtualized network, allowing interaction with user programs (Figure 3). To build application specific network services which involve packet processing, set particular parameters of the NE, and to facilitate other functions NEs may have in a network, NEs have the ability to deploy Application Components (ACs). ACs implement primitives to communicate with NEs and may employ existing network technologies or application specific ones [8-10].

The NE uses technologies, such as Web services, to expose interfaces on the Internet. Various applications can interact simultaneously with the NE through its interfaces, which are exposed as Grid services or as software objects in a WMS. As such, each application is capable to optimize the behaviour of the NE accordingly. During application development, NEs appear as objects (NC) in the development environment. During run-time, our model, as well as state of the art technology, allows dynamic extension of the set of NEs the applications interacts with.

By virtualizing network elements as software objects in the application domain, it becomes possible to control networks using software. Hence, application domain software can be used to program and automate the behaviour and management of network services.

III. IMPLEMENTATION

We developed a proof of concept to gain insight in the technical challenges involved in the virtualization of network elements and network control from Grid workflows. For the implementation we reuse and extend existing Grid software, such as the Globus Toolkit 4 [11]. The implementation of the proof of concept consists of two parts, which later are integrated to one solution.

The first part implements an extension of WS-VLAM scientific workflow management system with operations to reserve and control network resources. We integrate a programming interface of the virtualized network elements with the programming interface that WS-VLAM provides to Grid applications to manage computational resources. This adds network service management to Grid applications, which use the WS-VLAM application programming interface.

The second part implements a Network Operating System (NOS), which acts as an access point to the network and its resources. In principle, the network management system can implement the task of the NOS. However, network management systems are designed towards the needs of network operators, while we are interested in the challenges to expose the network services to applications. Therefore, the NOS interface (Figure 4, 4) is modelled after Grid broker to facilitate the integration with Grid software.

A. WS-VLAM Grid Workflow Management System

WS-VLAM aims to provide and support coordinated execution of distributed Grid-enabled components combined in a workflow. This workflow management system takes advantage of the underlying Grid infrastructure and provides a flexible high-level rapid prototyping environment. WS-VLAM consists of a workflow engine that resides within a Globus Toolkit 4 container on a server side and a workflow editor on a client side. A graphical representation of a workflow is created in the workflow editor (1) and forwarded to the engine to be scheduled and executed on the Grid (Figure 4). The engine consists of two WSRF [12] services: the Resource Manager (RM) and the Run-Time System Manager (RTSM). The Resource Manager is responsible for discovery, selection and location of resources to be used by a submitted workflow. The Run-Time System Manager performs the execution and monitoring of running workflows (2). In WS-VLAM all distributed application are represented as a workflow in the form of a data driven Direct Acyclic Graph where each node represents an application or a service on the Grid. The WS-VLAM workflow is data-driven; workflow components are connected to each other with data pipes in a peer-to-peer

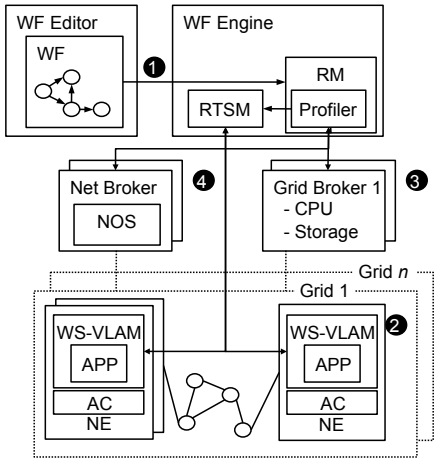

Figure 4. The architecture to control network resources as software objects in WS-VLAM.

manner via input/output ports. When new data arrives to an input port of a workflow component the data are processed by application logic and the result is transferred further via an output port.

Workflow components can be either developed using WS-VLAM library API or can be created by wrapping existing 'legacy' applications into a container. When an application is wrapped the workflow system takes care of state updates and provides an execution environment that allows the manipulation of various aspects of the application.

Workflow components are able to communicate by data streams with each other, with the workflow management system, at run-time with users by means of parameter interface. Each workflow component can have a number of parameters that can be set by a user of workflow management system to control the execution or by the component itself to signal or report some state. For example, the parameter interface can be used by a workflow component to request additional resources like a higher-speed network connection from the workflow management system. The initial WS-VLAM design implied the selection and control of computation resources only – what is typically provided by Grid middleware.

B. The Network Operating System and Integration of the Network with WS-VLAM

Because network details are not relevant to most applications, network only expose end-to-end transport services. Exposing control over network services from the application domain enables WMS and applications to develop and manage application-specific services. However, it also imposes complex provisioning issues to the application domain. To address network related provisioning, we introduce a Network Operating System (NOS), which acts as a single point of access to the WMS and hides network-specific complexities, such as provisioning of network elements in a consistent manner. The role of the NOS is to:

1. Manage user/application access to network resources (e.g. authentication, allocation, release),
2. Ensure fair usage (e.g. resource budgets, prioritization, scheduling),
3. Prevent errors in network resource allocation (e.g. creating paths between not connected nodes, exception handling).

The NOS communicates with clients through dedicated control connections. On start-up, the NOS clients register itself to the NOS. The AC subsystem in NOS clients uses Streamline [8-10] to load an application-specific network service. We limit ourselves to the manipulation of routes, but Streamline enables the implementation and dynamic reconfiguration of complete routing protocols. The NOS is the entry-point for coordinated access and control of network services. On behalf of the application, it loads or modifies Streamline modules on the NEs. The NOS builds and maintains a network model by executing a discovery mechanism at Ethernet level through which it collects all the neighbours of connected clients.

1) Defining an Application-specific Network Service

A chain of packet processing modules (Figure 5) defines which packets are filtered and which the network behavior is applied. A number of such chains loaded on several NEs define an application-specific network service. In order to provision a network service, which may be as simple as a static path for a source and destination of the application, each chain needs to be provisioned. The NOS uses a distributed transaction monitor to execute the provisioning. When a load or modification of a NE fails, it rolls back the manipulations of all the NEs to keep the network in a consistent state. The NOS assigns each network service a protocol independent token, which is stored in the IPv4 option field. The token is used to authenticate and filter each packet according to the provisioned packet processing chains. This way, tokens associate network services with the traffic in which they are embedded.

2) Network Control from WS-VLAM

A Network Broker (NB) (4) encapsulates the NOS to provide the same function as grid brokers (3), i.e. capabilities to query, request and load network services (Figure 4). Because the resulting interface is similar to Grid brokers, it is straightforward to extend the WMS with an additional service to include discovery, allocation and provisioning of network resources and services via the NB.

```
(netfilter_fetch_in) \
  >(fpl_tbs,expression="TOKEN") \
  >(fpl_ipdest,expression="DST_IP") \
  >(skb_transmit)
```

Figure 5. A Streamline request in which packets are taken from the Linux Netfilter [13] hook, then filtered by token and the IP destination overwritten.

The flexibility of network connectivity in application and workflow components depends on the usage of the WS-VLAM libraries. If the WS-VLAM API is used, WS-VLAM automatically handles establishment of connections and other technical Grid issues and provides the application developer the resulting data pipes created to its peer (e.g. in the form of C++ or Java IO stream). In WS-VLAM, network connections are implemented with Globus sockets and utilize only TCP (Figure 6). If (legacy) applications are wrapped in WS-VLAM however, connection handling remains the responsibility of the application. Although WS-VLAM does not control such

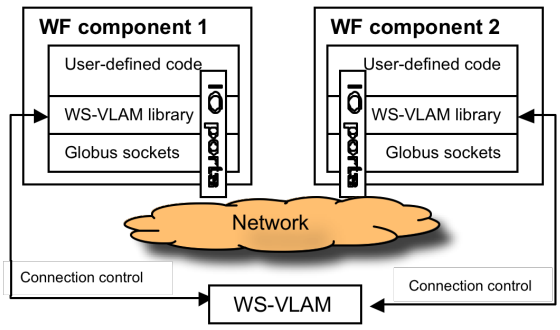

Figure 6. Connection handling for WS-VLAM workflow components.

connections directly, it does provide applications the means to control network services. In this case, UDP and other protocols can also be supported.

The network broker receives requests from the WS-VLAM engine via the profiler (see Figure 4). On successful execution of the request, the network broker returns a token to WS-VLAM that associates the network service with the request. WS-VLAM then attaches the token to the data sent by the application. On its turn, the NOS can identify the token (in the IPv4 packet) and apply the associated network service. However, the node on which the application runs has to attach the tokens to the packets. When the API is used, WS-VLAM provides tokenization of application traffic. When the application cannot use the API calls, because the source code cannot be modified for example, WS-VLAM will use socket interposing mechanisms [14] to tokenize the traffic.

IV. Experiments and Results

We developed a test bed and performed a series of experiments to evaluate the proof of concept and the feasibility of our approach. The experiments concentrate on critical aspects of the proof of concept in various application scenarios. We look in special at scenarios in which the network has to be controlled at run-time, i.e. a continuous loop of monitoring and adaptation to achieve or stay in an optimal configuration. Reservation-time scheduling of network services is trivial; the network is setup just before execution using the same mechanisms. We illustrate the basic steps of resource reservation, allocation and execution of an application in two cases. In one case, the network needs complete reconfiguration, because the data sources change. In the second case, the application requests better connectivity parameters. First, however, we provide a summary of our experimental test bed.

A. *Experimental test bed and performance of programmable nodes*

All the nodes in the test bed run Globus Toolkit 4 and also the programmable network software (Streamline) at kernel level. The network broker and WS-VLAM run on separate machines over a control network. Figure 7 shows our test bed in which nodes are interconnected through two networks, as follows: the default network uses a shared 100Mbit switch and the second network uses an IDXP2850 network processor unit programmed to route IP packets at 1Gbps.

TABLE 1 shows the overhead introduced by our packet manipulation components in Streamline within the kernel of each node. We used IPerf to exchange TCP and UDP traffic between two nodes (over 1Gbps network) in both cases: with and without Streamline. With Streamline, the jitter is larger and the total throughput is lower than without Streamline. However, this overhead is acceptable for our experiments.

TABLE 1
Packet manipulation performance

	Throughput (Mbps)	Jitter (ms)
Streamline		
TCP	317	
UDP	509	0.026
Non-streamline		
TCP	442	
UDP	798	0.009

B. *Run-time request of new data sources scenario*

This scenario illustrates how Grid applications can manipulate network services from WS-VLAM. Although changing paths may involve both computational and networking resources, when a new aggregation point needs to be chosen for example, for simplicity we look only into the networking resource brokerage.

Sensor networks and Grids are different systems in reality, but here we assume that a sensor network can provide a Grid service, which can be wrapped as workflow component. We implemented a sensor workflow component in WS-VLAM, which generates random UDP data using IPerf. The workflow component is used to simulate a realistic scenario in which the application needs to switch to a different data source, such as a radio telescope, for example, to continue running.

In this experiment, two WS-VLAM workflow components (W_1 and W_2) are re-directed to a single consumer (R) by an application that processes the data (Figure 7). When the application chooses to request the data of a different sensor, WS-VLAM requests NOS to release the current resources to (W_1) and to set up a new path from node (W_2) to (R). To make a clear distinction between W_1 and W_2, IPerf was used to generate data with a bandwidth of 10Mbit for W_1 and 30 Mbit for W_2, which enabled us to visually verify switching from data sources.

In our current implementation completing a switch from W_1 to W_2 takes less than three seconds. The mechanisms to load the new behaviour on the network elements, such as a two-phase commit protocol, introduce this delay. In the worst case, a node that does not reply for any reason causes the commit process to wait for the time-out until failure. In the future, we expect to improve the performance of reconfiguring network elements.

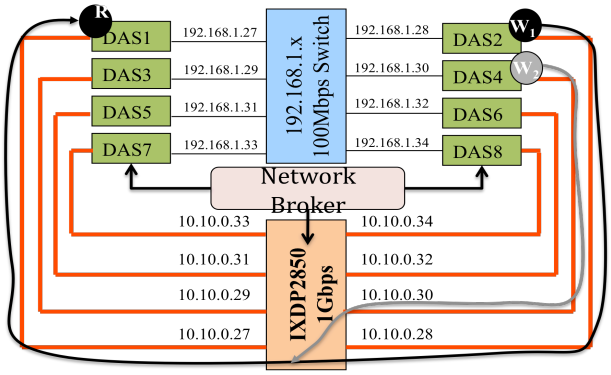

Figure 7. Test bed setup and programmed routes in the first experiment. The network broker accesses the nodes over a separate control plane.

C. *Run-time request of better connectivity scenario*

Figure 9 shows a screenshot from the workflow manager with multiple workflows. The workflow manager starts

workflows one by one: 1, 2, 3, 4. When the network performance (throughput) measured by an application decreases below a certain threshold, the application will request better connectivity from WS-VLAM. WS-VLAM will then offload the resources of the requesting application from the 192.168.1.x network onto the 10.10.0.x network (e.g. path 4 moves to the 1Gbps network), yielding improved performance.

The performance of the experimental application on the test bed is illustrated in Figure 8. Due to the shared 100Mbps, the per-path performance decreases while more paths are established and exchange data traffic at maximum. The switch offers one single network service: best effort. The application running in the workflow (*bwMeter* in Figure 9) measures the throughput and when it reaches a programmable threshold, it requests more resources for the current configuration to WS-VLAM. Next, NOS receives a demand for better paths and decides to create alternative paths over 1Gbps network. Consequently, we see that the throughput increases (*bwMeter* in the second part of Figure 8).

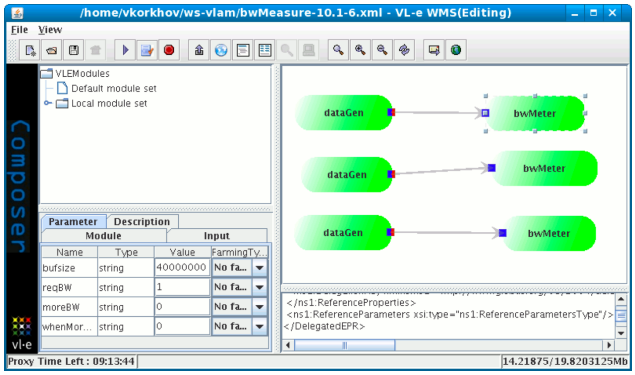

Figure 9. Screenshot of the workflow editor, which shows WS-VLAM workflow where multiple producers (dataGen module) and consumers (bwMeter) are connected.

Figure 11 illustrates automatic switching of network paths according to application requests. First, the application is started on a network that provides the throughput of 12 MB/s (section A in Figure 11). Later another application starts using the same network link which results in decreased network performance of the current application (section B). At some point application needs to temporarily increase the throughput and acquire more bandwidth (e.g. for a scheduled bulk data transfer). This happens in the section C: the application switches itself to use another faster network. After the needed data transfer action has been performed the path on the fast network can be released and the initial network is used again (section D).

V. Discussion

The experiments with our proof of concept show the feasibility of network control from Grid workflow management system. Traditionally, applications only deal with end-to-end transport services in the network. Our approach changes the way applications can deal with network details. Amongst other things, we need to investigate programming

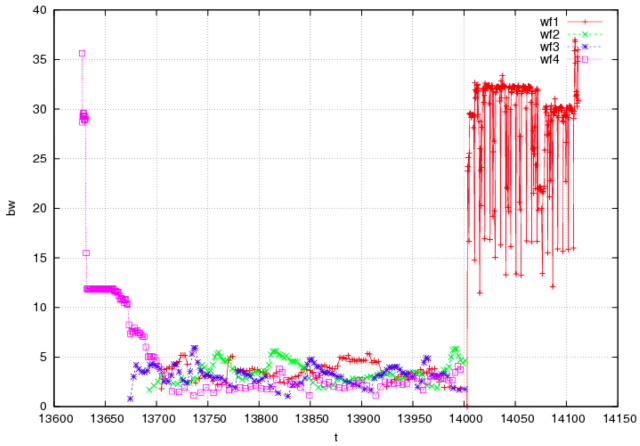

Figure 8. Experimental evaluation of test bed (bw in MB/s and t in seconds).

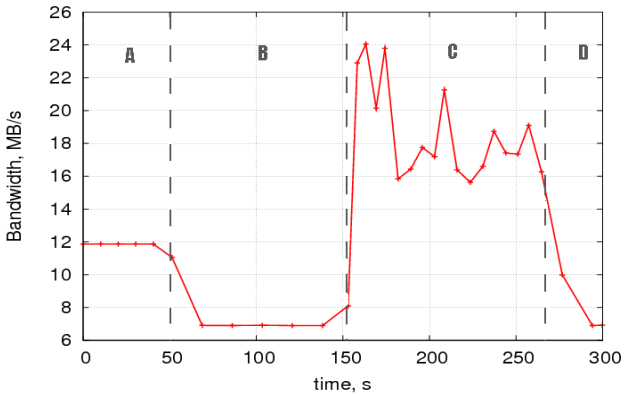

Figure 10. Bandwidth measurements while the application automatically controls throughput.

models to develop network services and determine the scalability of our approach. While the proof of concept only supported path manipulation, in the future we plan take full advantage of Streamline to develop complex application-specific routing protocols. Because this is our first attempt, however, many issues still remain.

We experienced the most difficulties in supporting application-specific UDP and TCP traffic manipulation for legacy applications. While supporting UDP is straightforward, supporting TCP is difficult because of the many complex mechanisms that are part of the protocol. Therefore, the behaviour of the protocol has to be understood (and in some cases worked around) while implementing application-specific services.

We noticed that when using the IXDP2850 network processor the jitter increases significantly in the TCP flows. This might be caused by side effects of the code, which processes and rewrites the IP packets. In general, jitter is caused when application-specific code is executed and could be improved taking into account the multi-core architectures. Although decreased throughput is partly caused by the programmable network, a significant performance gain is expected when more efficient mechanisms to capture traffic are implemented.

Our approach to match and schedule the network after the Grid broker allocated the computing nodes sufficed in our

proof of concept, but might be inefficient in larger systems. Moreover, when an application requests new computational resources, network resources need to be rescheduled. Better approaches need to be developed and evaluated in real Grid environments.

In this paper, we have not addressed the issues of managing resources over multiple domains. As with grid brokers, we believe that every network can have its own network broker and NOS. But, to effectively reserve, load and connect network services to applications over multiple network brokers remains a challenge. At least at the network level, tokens in IP packet could be used for multi-domain authentication and for associating traffic to applications. Although technically feasible, the biggest hurdle is expected to be the administrative efforts to allow network resource reservation and control to span over multiple domains.

VI. Related Work

Coordinated configuration of Grid and network resources is difficult in practice [15]. The effort can be justified in large e-Science applications, however, because it might be easier to run experiments over multiple Grids rather than claiming a large portion of resources in one Grid. But, to achieve acceptable performance over multiple Grids, communication links between nodes the Grids should be optimized. In most cases, this means configuring dedicated communication paths between the Grids. Progress in grid network research can be characterized by this goal.

Some attempts have been made to incorporate network resource orchestration into workflow management systems. WINNER (Workflow Integrated Network Resource Orchestration) from Nortel Network Labs [16] proposed a way to integrate network resources with WS workflows; DRAC [17] network services are leveraged here for allocation and information in network resource orchestration. However, this project was mostly oriented to business workflows and seems to be not maintained at the moment. In the scope of scientific Grid workflow management systems we are not aware of initiatives to integrate network as a fully featured manageable resource on the workflow and application level.

Several Grid projects attempt to extend control over network resources into the Grid toolset. So far, the efforts focused on reservation, traffic engineering of network circuits and improving the performance of network protocols [18-23], but did not expose the new capabilities to workflow management tools where they can be easily accessed and used by applications and end-users.

In our architecture, we have looked at the virtualization of programmable network elements as software objects in workflow management systems. We believe that (1) a workflow management system is the right place to control and manage networks and interface with applications and (2) programmable network technologies are sufficiently well understood to be applied in Grid networks of the future. For example, the next generation of the DAS-4 [24] super computer will include FPGAs in the network fabric. To support our approach in current grid networks, though with the flexibility offered by existing network technologies, we need to take advantage of the progress made in Grid networks. Here, we summarize some of the state-of-the-art Grid networks.

The G-Lambda project successfully conducted the first experiments on coordinated scheduling of network and Grid resources [20, 25]. In G-Lambda's approach, users submit a job to a global Grid resource scheduler, which on its turn allocates resources by negotiating with individual computational and network resource managers. G-Lambda focuses on provisioning optical light-paths between Grids, using MPLS for path provisioning.

UltraLight aims to provide a dynamically configurable network to support high-performance, distributed data-processing application for the high-energy physics community [19]. In their approach, the network is considered a resource and is closely monitored. In addition, the system can take advantage of monitoring information about network state to optimize network resource usage.

The e-Toile project introduces active networks into the Grid domain [26]. By doing so, they go further than end-to-end path reservation and also allow applications to inject code into the network. Their active networks framework TAMANOIR [27] places active nodes at the edges of Grid networks, which can be a platform for adding new and innovative network services.

In non-Grid related network research, other efforts propose to make the management layer programmable [28, 29]. In this approach, computer programs define policies and network services and control switches and routers that implement flow routing. Such an approach fits well with our goals, because they support application-specific network services and are backwards compatible with current network technologies.

VII. Conclusion and Future Work

Network performance is crucial to a class of communication intensive applications loosely defined under the term Sensor Grids. Such applications may run in Grids, because it is too costly or unfeasible to develop dedicated systems. Unfortunately, despite the advanced management of computational and storage resources, Grids lack network resource management. To support communication intensive applications or applications with specific network demands in Grids, we introduced a novel architecture. In this architecture, network elements are virtualized as software objects in the application domain and managed by a Grid workflow management system. Moreover, we have built a proof of concept using an existing workflow management system (WS-VLAM) and performed a series of experiments to demonstrate the feasibility of our approach.

Some Sensor Grid applications require immediate access to large amount of resources and control over networks triggered by an event. For example, dangerous water levels detected at dike may result in a large-scale simulation for risk assessment. We have not yet implemented dynamic reservation of shared

network resources for urgent computing. Although, we can extend WS-VLAM and the network manager to support application priorities, such a feature also needs to be supported by Grid brokers.

While we believe the presented work is a promising approach to support communication intensive applications and to build Sensor Grids, it also raises questions about scalability and security. How can Grid applications control potentially tens of thousands of nodes? Can we use Grid workflow managers to automate parts of the implementation we did by hand? What are the implications for network management and the risks to network operators? In other words, to develop and evaluate practical implementation of the network as a resource in Grids is a topic for further research.

REFERENCES

[1] R. J. Meijer and A. R. Koelewijn, "The Development of an Early Warning System for Dike Failures," in *1st International Conference and Exhibition on WATERSIDE SECURITY* Copenhagen, Denmark, 2008.

[2] N. Kruithof, "E -Vlbi Using a Software Correlator," in *Grid Enabled Remote Instrumentation*, 2009, pp. 537-544.

[3] "CERN". [Online]. Available: http://public.web.cern.ch/public/ [Accessed: 11 November, 2009].

[4] H. Lim, et al., "Sensor grid: integration of wireless sensor networks and the grid," in *Local Computer Networks, 2005. 30th Anniversary. The IEEE Conference on*, 2005, pp. 91-99.

[5] E. Rosen, A. Viswanathan, and R. Callon, "Multiprotocol Label Switching Architecture," *RFC3031,* January 2001.

[6] V. Korkhov, D. Vasyunin, A. Wibisono, V. Guevara-Masis, and A. Belloum, "WS-VLAM: Towards a Scalable Workflow System on the Grid " in *Workshop on workflows in Support of Large-Scale Science (WORKS 07) In conjunction with HPDC 2007* Monterey Bay, California, 2007.

[7] R. J. Meijer, R. J. Strijkers, L. Gommans, and C. de Laat, "User Programmable Virtualized Networks," in *Proceedings of IEEE International Conference on e-Science and Grid Computing*: IEEE Computer Society, 2006.

[8] J. Elischer and A. Cobbs, "FreeBSD Netgraph pluggable network stack". [Online]. Available: http://www.freebsd.org/ [Accessed: 10 August, 2009].

[9] H. Bos, W. d. Bruijn, M. Cristea, T. Nguyen, and G. Portokalidis, "FFPF: Fairly Fast Packet Filters," in *OSDI*, 2004.

[10] R. Morris, E. Kohler, J. Jannotti, and M. F. Kaashoek, "The Click modular router," *SIGOPS Oper. Syst. Rev.,* vol. 33, pp. 217-231, 1999.

[11] I. Foster, "Globus Toolkit Version 4: Software for Service-Oriented Systems," in *Network and Parallel Computing*, 2005, pp. 2-13.

[12] "OASIS Web Services Resource Framework (WSRF)". [Online]. Available: http://www.oasis-open.org/committees/wsrf/ [Accessed: 16 March, 2010].

[13] "Linux Netfilter". [Online]. Available: http://www.netfilter.org [Accessed: 17 August, 2009].

[14] M. Cristea, et al., "Supporting Communities in Programmable Networks: gTBN," in *IFIP Integrated Management 2009* New York 2009.

[15] S. P. Zwart, et al., "Simulating the universe on an intercontinental grid of supercomputers," *Submitted to IEEE Computer,* 2009.

[16] P. Wang, I. Monga, S. Raghunath, F. Travostino, and T. Lavian, "Workflow Integrated Network Resource Orchestration," presented at GlobusWorld, Boston, 2005.

[17] R. K. F. Travostino, T. Lavian, I. Monga, B. Schofield, , "Project DRAC: Creating an application-aware network," *Nortel Technical Journal,* February 2005.

[18] E. Grasa, G. Junyent, S. Figuerola, A. Lopez, and M. Savoie, "UCLPv2: A network virtualization framework built on web services," *IEEE Communications Magazine,* vol. 46, pp. 126-134, Mar 2008.

[19] H. Newman, et al., "The UltraLight Project: The Network as an Integrated and Managed Resource for Data-Intensive Science," *Computing in Science and Engg.,* vol. 7, pp. 38-47, 2005.

[20] S. R. Thorpe, et al., "G-lambda and EnLIGHTened: wrapped in middleware co-allocating compute and network resources across Japan and the US," in *Proceedings of the first international conference on Networks for grid applications* Lyon, France: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2007.

[21] F. Travostino, *Grid Networks: Enabling Grids with Advanced Communication Technology*: Wiley, 2006.

[22] K. Yang, X. Guo, A. Galis, B. Yang, and D. Liu, "Towards efficient resource on-demand in Grid Computing," *SIGOPS Oper. Syst. Rev.,* vol. 37, pp. 37-43, 2003.

[23] B. Berde, A. Chiosi, and D. Verchere, "Networks meet the requirements of grid applications," *Bell Labs Technical Journal,* vol. 14, pp. 173-184, 2009.

[24] H. E. Bal, "DAS-4: Prototyping Future Computing Infrastructures". [Online]. Available: http://www.nwo.nl/projecten.nsf/pages/2300154150_Eng [Accessed: 16 March, 2010].

[25] A. Takefusa, et al., "G-lambda: Coordination of a Grid scheduler and lambda path service over GMPLS," *Future Generation Computer Systems,* vol. 22, pp. 868-875, 2006.

[26] A. Bassi, et al., "Active and logistical networking for grid computing: the e-Toile architecture," *Future Generation Computer Systems,* vol. 21, pp. 199-208, 2005.

[27] F. Bouhafs, et al., "Designing and evaluating an active grid architecture," *Future Generation Computer Systems,* vol. 21, pp. 315-330, 2005.

[28] N. McKeown, et al., "OpenFlow: enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev.,* vol. 38, pp. 69-74, 2008.

[29] M. Casado, et al., "Ethane: taking control of the enterprise," *SIGCOMM Comput. Commun. Rev.,* vol. 37, pp. 1-12, 2007.