Towards a Light-weight Bag-of-tasks Grid Architecture

Ilija Bašičević, *Member, IEEE*, Nenad Četić, Miroslav Popović, *Member, IEEE,* and Momčilo Krunić

***Abstract* — The paper presents the application of SIP protocol in the context of bag-of-tasks grid architecture. The SIP protocol has been used in the realization of the execution management service. The main idea is the use of stateful SIP proxy as a request broker. The paper provides a description of the concept, and the prototype system that has been built, as well as the calculation of estimated performance level and its relation to maximum RTT of grid system. The main advantage of this light-weight grid architecture is the reuse of a mature infrastructure. A short overview of some approaches to the mathematical modeling of computer grids is included.**

***Keywords* — Session Initiation Protocol, grid computing, bag-of-tasks, execution management service, job scheduling.**

I. Introduction

COMPUTING grids have been researched for already more than a decade. Such systems are used for different scientific and commercial purposes, like weather analysis and forecasts, fusion modeling, or commercial data centers. There is a variety of resources that can be shared, ranging from disk space to processing power. The basic premise is the creation of interoperable, reusable and portable components. The next step is solving many issues related to security, quality of service, monitoring and management of components and services. OGSA [1] is a well known grid architecture, developed by Global Grid Forum (today Open Grid Forum). It is based on the concept of web services, among others. We use it here to define the context for the proposed architecture. OGSA defines a set of functions which compose a grid. This paper's focus is on the following OGSA functions: execution management services, notification, and security.

Paper received March 24, 2015; accepted April 29, 2015. Date of publication July 15, 2015. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Jovan Đorđević.

This paper is a revised and expanded version of the paper presented at the 22th Telecommunications Forum TELFOR 2014.

This work was partially supported by the Ministry of Education and Science of the Republic of Serbia under the project III 44009-2.

Ilija Bašičević, Faculty of Tech. Sciences, University of Novi Sad, Serbia (email: ilibas@uns.ac.rs).
Nenad Četić, Faculty of Tech. Sciences, University of Novi Sad, Serbia (email: nenad.cetic@rt-rk.com).
Miroslav Popović, Faculty of Tech. Sciences, University of Novi Sad, Serbia (email: miroslav.popovic@rt-rk.com).
Momčilo Krunić, RT-RK, Institute for Computer Based Systems LLC, Novi Sad, Serbia (email: momcilo.krunic@rt-rk.com).

doi:10.5937/telfor1501049B

II. Session Initiation Protocol

SIP [2] is a signaling protocol targeted for multimedia communications over the Internet. It reached maturity with version 2, published in 2002. The protocol syntax is based on HTTP. SIP is the most important open protocol in this area today. It covers functions such as location service, session negotiation, establishment, modification, and termination. In VoIP, which is the primary use of SIP today, this session is a multimedia communications session. We discuss here the possibility of extending the notion of SIP session to jobs in a grid network. We note here that SIP protocol was initially developed as a general rendezvous protocol, and that VoIP later became its main application. The same functions that SIP covers are required in grid environment. Based on those functions we can build more complex grid services, in particular execution management.

III. Proposed Grid Infrastructure

A. The Grid Environment

User-defined work is in OGSA referred to as a job. Jobs range from simple to complex, examples of which are composite jobs and workflows. Grid should include job schedulers that work based on the priority of jobs and allocation of resources. OGSA is a service-oriented architecture where services are in a peer relationship. In this paper, we focus on the middle tier of OGSA architecture, where abstraction and virtualization of OGSA result in so called capabilities. An example of a capability is execution management. Important issues that are handled by this capability are:

- finding candidate locations for execution of a user defined job
- selecting execution location
- preparing the selected location for execution
- initiating the execution
- managing the execution.

B. Applying SIP in the Grid Case

The basic scenario of job scheduling is presented in Fig.1. A client requesting a service sends a SIP INVITE message to SIP proxy acting as a broker. SIP proxy fetches the list of candidate locations from location service - using REGISTER message. INVITE is forwarded to each location in the list. This is the application of request forking concept. Proxy collects responses from candidate locations, and selects the optimal one, according to defined criteria. The selected response is forwarded to the client. Requests to other locations are canceled.

The scenario presumes that candidate locations have been registered at location service. Each candidate location runs a SIP user agent. The client requesting the service also runs a SIP user agent. The advantage of this architecture is that location service (registering and querying for the list of locations) as well as rendezvous protocol (client requesting service from proxy, proxy dispatching the request to candidate locations, and in the last step handshake between the client and the selected location) are already implemented as part of SIP user agent, registrar and proxy. The only modification to SIP software layer is the procedure for selection of location among candidates that responded to the forwarded request.

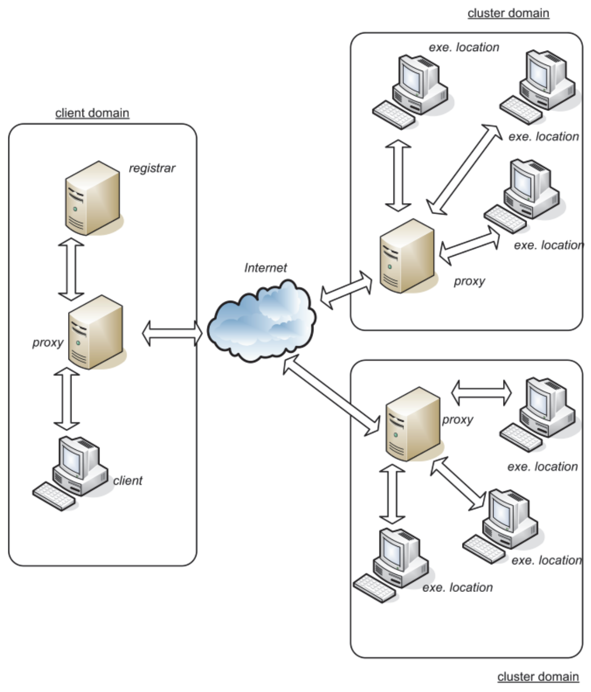

Fig. 1. Proposed grid architecture.

The selection procedure is based on optimum criteria and depending on the design of SIP proxy can be implemented in the higher layer (using software hooks) which is technically a more reliable solution or by modifying the proxy software itself. In any case, the modifications are limited and localized. The INVITE message carries, besides typical SIP headers prescribed by standard, an additional body with the description of requested task. The description is textual, written in JSDL [3] language. JSDL [4] is a standard language for that purpose, proposed by Global Grid Forum. We note that in VoIP use of SIP, INVITE message carries a SDP [5] description of offered session in message body, so the only deviation from VoIP scenario (with respect to messages) is the use of JSDL text body instead of SDP. The next step after request distribution is as follows. The client requesting the service receives 200 OK, containing the location IP address. The client sends an ACK request to the selected job execution location. If needed, the client uses a re-INVITE mechanism to send additional data. In the re-INVITE mechanism, the client uses the IP address of server to send an INVITE message (carrying additional data in the body of message) directly to the server, without any intervention of proxy.

The architecture is proposed for bag-of-tasks applications [6] in which there is no communication between independent tasks. A basic requirement of the majority of computational activities is security. OGSA presumes the use of WS-Security protocols for purposes of authentication, authorization and message protection. Both application level and transport level (e.g. TLS) mechanisms are required in grid. SIP provides solutions, many of which are reused from HTTP. SIP includes mechanisms for authentication (both peer-to-peer, and peer-to-proxy) and allows encryption of parts of the message (most notably message body). Encryption of the whole message is not suitable, because SIP infrastructure elements along the message path (proxies) need access to some of the message headers. For this reason, SIP message syntax already provides for the transmission of user credentials, which is of interest in the grid case.

C. *The realization of prototype system*

For the realization of SIP components we have used mjSIP [4]. It is an open source SIP protocol stack, developed in the Java programming language. The prototype system has been developed in the Eclipse development environment, as a set of plugins:

- client application that requests service
- server application that provides service (execution location)
- proxy application that acts as request broker (execution management service)
- JSDL plugin.
- SIP plugin, which is in fact mjSIP library, with certain modifications.

The most important class in JSDL plugin is JSDLParser, which maps JSDL data to Java classes (unmarshalling) and vice versa (marshalling). In SIP plugin UserAgent class, the call method has one argument - job description. OnCallIncoming method of the same class is invoked on the server side, and includes unmarshalling of received JSDL job description and verification that requested resources exist on the server. Upon reception of 200 OK from the server, OnCallAccepted is invoked on the client side. For parsing of MIME bodies in SIP messages, MIMEUtility class has been developed. We have tested a limited case, in which there is a relatively small number of candidate servers, all present in a local network, and belonging to the same SIP domain.

IV. Performance of a SIP-based Grid System

The performance of the proposed system depends on the delay introduced by SIP client-server negotiation. In the grid case, there will be a large number of clients (the order of magnitude is hundreds or even thousands), dispersed in many SIP domains. We assume that there would be a shallow hierarchy, in which local location service in clients' domain would return a list of candidate clusters. The hierarchical structure is used in the majority of existing grid systems today. In a general case each candidate cluster would belong to a different SIP domain. Local location service in each candidate domain would map the address to a set of candidate locations in that domain. Each candidate location that is available and provides a requested service would respond with 200 OK, and that message would be routed over the proxy in a candidate cluster to the proxy in the client domain. The total time for distributing the request in case of prototype system is:

$$T_{total} = T_{client\_to\_local\_proxy} + T_{location\_query\_in\_client\_domain} + T_{forwarding\_requests} + T_{collecting\_answers} + T_{selecting\_candidate} + T_{local\_proxy\_to\_client} \quad (1)$$

In all calculations, we assume that there is an available server at the moment, thus no waiting in queue is required. It can be assumed that:

$$T_{client\_to\_local\_proxy} + T_{local\_proxy\_to\_client} = RTT_{in\_client\_domain} \quad (2)$$

If we consider now the hypothetical grid, $T_{client\_to\_local\_proxy}$ is exactly the same as in the prototype, $T_{location\_query\_in\_client\_domain}$ is approximately very close, and can be assumed to be the same.

Similarly to 2, it can be assumed that:

$$RTT_{inter\_domain} = T_{proxy\_to\_cluster\_proxy} + T_{cluster\_proxy\_to\_proxy} \quad (3)$$

$$T_{cluster\_proxy\_to\_server} + T_{server\_to\_cluster\_proxy} = RTT_{in\_cluster\_domain} \quad (4)$$

$$RTT_{in\_cluster\_domain} = RTT_{in\_client\_domain} \quad (5)$$

$T_{location\_query}$ in both cases, cluster and client domain, has the following components.

$$T_{location\_query} = RTT_{intra\_domain} + T_{processing\_query\_request} + T_{processing\_query\_response} \quad (6)$$

A maximum message size when using SIP over UDP is 65536 bytes. But the standard prescribes that SIP implementation should use a congestion controlled transport protocol (RFC 2914), such as TCP, if the size of request is within 200 bytes of MTU, or if it is larger than 1300 bytes, and MTU is unknown. The response should be sent using the same connection (if the request was sent using a connection oriented protocol, and the connection is still open), otherwise the value of Via header determines the transport that would be used.

It can be estimated that depending on the values of other message headers, REGISTER response of 1500 bytes can carry around 24 Contact headers. A message of 1300 bytes would carry around 20 Contact headers. That allows the realization of location query over UDP, without fragmentation, for a grid of 20 clusters, with 20 candidates in each cluster.

In the hypothetical grid case,

$$T_{total} = 3{*}\,RTT_{in\_client\_domain} + 3{*}RTT_{in\_cluster\_domain} + \max\{RTT_{inter\_domain}\} + T_{processing\_query\_request\_in\_client\_domain} + T_{processing\_query\_response\_in\_client\_domain} + T_{processing\_query\_request\_in\_cluster} + T_{processing\_query\_response\_in\_cluster} + T_{selecting\_candidate} \quad (7)$$

In our prototype system we have used XML bodies in REGISTER messages to convey the descriptions of server features. It is the same concept that is used for conveying CPL scripts in IP telephony [7]. This allows us to implement logic for matching of required resources/features with server descriptions – on the proxy side. The measurements have shown that a rise in $T_{selecting\_candidate}$ does not follow linearly a rise in the number of candidates. This is a consequence of the use of general purpose operating system and Java virtual machine. We present the cumulative distribution function of $T_{selecting\_candidate}$ in Fig 2. For comparison, CDF of normal distribution is given. Note that:

$T_{selecting\_candidate} = 4.196$ msec
$\sigma = 1.169$ msec
$T_{selecting\_candidate\_in\_bare\_mjSip} = 1.647$ msec

When compared to 7 msec which is $RTT_{in\_client\_domain}$, measured in our network the $T_{selecting\_candidate}$ is significant. But in a hypothetical grid, the $\max\{RTT_{inter\_domain}\}$ is potentially the dominant component, far outreaching $T_{selecting\_candidate}$. As it is well known, RTT for intercontinental and satellite links can reach 400 ms. On the other hand, the time components for processing query requests and responses are smaller than $T_{selecting\_candidate}$.

In some implementations, including mjSIP, registrar and proxy are collocated. In that case ($RTT_{in\_client\_domain} + RTT_{in\_cluster\_domain}$) should be subtracted from T_{total}. Also, the time components for processing query requests and responses should be lowered, because only the most relevant information is carried in function calls – there is no overhead in building/parsing messages.

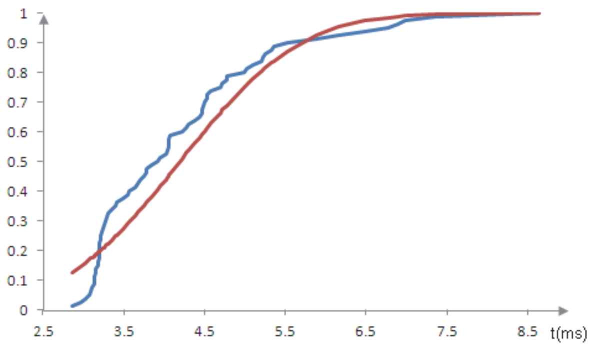

Fig. 2. CDF of duration of candidate selection procedure.

V. The Mathematical Model of SIP-based Grid Prototype

In order to successfully run and make use of a distributed system, the system provider needs to know several operational parameters, such as the probability that k servers are busy, average queue length, average waiting time, etc. To obtain these values, the system should be mathematically modeled and possibly simulated. In this section we list formulas for operational parameters of queuing systems that can be used for modeling computing grids. Some of the systems have not been solved analytically, or the formulas are not practical for use. Therefore we list two approaches for the numerical solution of these systems.

When modeling out SIP grid system, if we have in mind the general case, the system can be modeled as an M/G/n system in the Kendall notation, see Fig 3. This applies to a bag-of-tasks system. For a more general grid system (e.g. that supports workflow) the model is based on a servicing network. Typically, each node in the network is modeled as M/M/n and the communication channels between nodes as M/M/1 systems. This is sometimes referred to as an M/M/n + M/M/1 model.

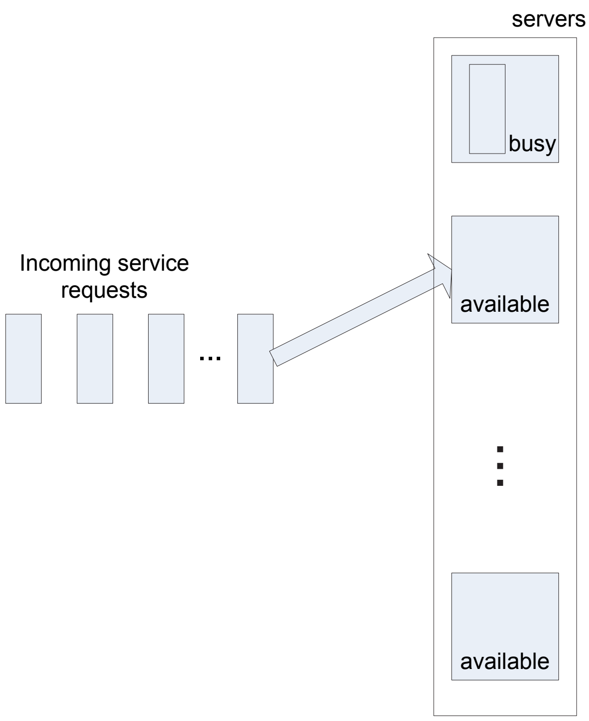

Figure 1. Queuing system with n servers and one incoming queue of service requests.

Returning to our system, the arrivals are Markovian and can be modeled as a Poisson process. Service times have a general distribution. There are n servers in the system. A special case of this system is M/G/1, a system with one server. Pollaczek [8] provided the formulas for M/G/1 system in the first half of the 20th century. The formulas have been interpreted by Khintchine [9], using the mathematical apparatus developed by Markov and Kolmogorov.

The Pollaczek-Khintchine formulas are given here:

$$W = \frac{A \cdot s}{2(1-A)} \cdot \varepsilon \tag{8}$$

$$W = \frac{V}{1-A}$$

Where,

$$V = A \cdot \frac{s}{2} \cdot \varepsilon = \frac{\lambda}{2} \cdot m_2$$

W is the mean waiting time for all customers, s is the mean service time, A is the offered traffic, λ is the arrival intensity, m_2 is the second moment of service time distribution, and ε is the form factor of the holding time distribution.

The mean value of the busy period is:

$$m_{T1} = \frac{s}{1-A} \tag{9}$$

The busy period is a period from the moment when all servers get busy until the moment when a customer leaves the system. On the other hand, the mathematical modeling of M/G/n system is still an open issue, and many performance metrics are still unknown.

If we consider our laboratory prototype, we can assume a constant service time, in which case the model is M/D/n. This can be assumed for a grid system that provides a single type of service, using a set of servers that are based on the same hardware and software platform. However, the characteristics of the operating system should be considered as well.

The special case of M/D/n is the system with one server and a constant service time. The M/D/1 type of system has been analyzed by Erlang, who found the waiting time distribution, and Fry, who contributed the state probabilities formula.

The mean waiting time for M/D/1 system is:

$$W = \frac{A \cdot h}{2(1-A)} \tag{11}$$

where h is a constant service time. The mean value of busy period for the same system is:

$$m_{T1} = \frac{h}{1-A} \tag{12}$$

State probabilities for M/D/n system are, in case that A < n:

$$p(i) = \left\{ \sum_{j=0}^{n} p(j) \right\} \cdot p(i,h) + \sum_{j=n+1}^{n+i} p(j) \cdot p(n+i-j,h),$$

$$i = 0,1\ldots \tag{13}$$

where $p(i,h)$ is the probability that there are i calls in time interval h. In order to obtain the explicit mathematical solution, generating functions are used [10]. The waiting time distribution is given by Crommelin's formula [11]:

$$P(W \le t) = 1 - \sum_{i=0}^{n-1} \sum_{k=0}^{i} p(k) \sum_{j=0}^{\infty} \frac{\{A(j-\tau)\}^{(T+j+1)n-1-i}}{\{(T+j+1) \cdot n - 1 - i\}!} \tag{14}$$

where A is offered traffic, and

$t = h \cdot T + \tau$

$0 \le \tau \le h.$

However this formula is not practical for calculation.

The realistic model of grid should include a queue. The state probabilities for a finite queue system M/D/1/k can be obtained in the following way. The system has k waiting positions and one server, that is (k+1) states. The formulas for k-1 states follow from the state probabilities for M/D/1 system:

$$p_{t+h}(i) = \{p_t(0) + p_t(1)\} p(i,h) + \sum_{j=2}^{i+1} p_t(j) \cdot p(i-j+1,h), i = 0,1... \tag{15}$$

That gives k-1 equations. Additional two equations are:

$$\sum_{j=0}^{k} p_j(i) = 1$$

$$A = 1 - p_k(0) + A \cdot p_k(k) \tag{16}$$

The last equation stems from the so called PASTA property of this type of system.

We mention here two approaches, one providing a simpler formula for an already existing solution, and the other one for obtaining performance metrics of a queuing system that has not been solved analytically.

Franx [12] showed that the waiting time probability for

M/D/n is:

$$P(W \le x) = e^{-\lambda(kD-x)} \sum_{j=0}^{kn-1} Q_{kn-j-1} \frac{\lambda^j (kD-x)^j}{j!} \,, \qquad (17)$$

$$(k-1)D \le x \le kD, Q_m = \sum_{0}^{m} q_i$$

where q_i is the probability of queue length i. It is obtained by solving a set of linear equations:

$$q_0 = \sum_{j-0}^{n} q_j \sum_{m=0}^{n-j} \frac{(\lambda D)^m}{m!} e^{-\lambda D}$$
$$q_i = \sum_{j=0}^{i+n} q_j \frac{(\lambda D)^{i+n-j}}{(i+n-j)!} e^{-\lambda D}, i > 0 \qquad (18)$$

The equations can be solved using the fast Fourier transform and a geometric tail approach [13].

Khazaei and Misic [14] show that the Laplace-Stieltjes transform W* of waiting time distribution W for M/G/n system is:
$W^*(s) = Q\,(1\text{-}s/\lambda)$, where Q is the queue length and equals

$$Q(z) = \sum_{k=0}^{n-1} \pi_k + \sum_{k=n}^{2n} \pi_k z^{k-n} \qquad (19)$$

where π_k is the equilibrium probability distribution for the number of tasks present at arrival instants $\pi_k = \lim_{m \to +\infty} P(q_n = k)$ and q_k is the number of tasks present in the system at the moment of m-th arrival.
The steady state probabilities π_k are obtained numerically from balance equations

$$\pi_i = \sum_{j=0}^{2m} \pi_j p_{ji}, 0 \le i \le 2m \qquad (20)$$

and normalization equation

$$\sum_{i-0}^{2m} \pi_i = 1\,. \qquad (21)$$

As already noted, M/M/n model has often been used for the modeling of grids in the existing literature, but recent research [15] highlights the advantages of M/D/n model. For the modeling of communications' channel, the authors use M/D/1 system. The simulation results are compared to two models (M/M/n + M/M/1 and M/D/n + M/D/1). The relative error for varying workloads on a single grid node in case of the M/D/n + M/D/1 model does not surpass 7%, while for the M/M/n + M/M/1 model it almost reaches 25%, depending on the workload.

VI. The improved architecture

SIP protocol provides means for further optimization. The solution is based on SIP presence service [16]. The foundation of presence service is the SIP event notification mechanism [17]. The mechanism is an implementation of publish-subscribe design pattern [18], adapted to SIP architecture. The presence service stores and distributes presence information. The service architecture recognizes three types of participants: server and two types of clients: presentities and watchers. We propose the use of subscribers, which is a specific type of watcher. Changes in presence information are distributed from presentities to subscribers, via servers and using event notifications. Based on presence mechanism a proxy application can be built that keeps evidence on the state of registered candidate locations: whether they are currently engaged in the service of clients or available for a new service request. The network traffic during request distribution in a cluster is reduced, compared to the case when the request is forwarded to all suitable servers. The system described above can implement standard HRN (Highest Response Next) [19] scheduling schemes too. When notification from a presence server about the availability of a server is received, the job queue is checked and if not empty, selection of a candidate is performed for the selected job. The procedure can be as simple as taking the job from the head of the queue, which implements the standard FCFS (First Come First Served) scheme, or the queue can be iterated to find a job that suits some criteria, as for example in the SJF (Shortest Job First) scheduling scheme.

VII. Related Work

Grid computing has already been researched for more than a decade. We mention here some of existing literature. To the best of our knowledge, a similar approach for use of SIP in this area has not been published.

GRAM job management system is a part of Globus Toolkit [20], a widely used grid platform, that has been developed since late 1990s. Globus supports hierarchical organization of grid. HTCondor, until 2012 known as Condor [21] is a workload management system that provides job queueing, resource monitoring, and resource management among other features. HTCondor-G is its scheduler, compatible with Globus - it can rely on GRAM for managing of remote jobs and GSI (Grid Security Infrastructure) for security. The advantage of SIP-based architecture compared to Globus and HTCondor is that it is a light-weight architecture, which requires a smaller number of software components, compared to the two. The only proposed grid architecture that makes use of SIP in available literature is [22], which is a wireless grid platform that uses SIP protocol and agents. SIP protocol is used for communication between users and agents, and between agents. Any modifications to SIP messages are not described (as JSDL used in message bodies in our proposal). The hierarchical architecture of grid and the use of SIP proxy for request brokering are specific to our proposal.

VIII. Conclusions and Future Work

The paper presents a SIP based bag-of-tasks grid architecture. SIP has been used as rendezvous protocol that provides infrastructure for distribution of requests, client-server negotiation, and registration of execution locations. A prototype that has been built using mjSIP protocol stack proves the viability of the proposed solution. The estimation of performance shows that request distribution time does not increase significantly with the increase of the size of the grid. The dominant component is maximum RTT which does not depend on the number of candidate locations in a cluster. On the other hand, the more dispersed the grid gets, the chance is that RTT of network paths could increase substantially in the case of slow networks, but grid administration can be used to exclude those paths and clusters. This reasoning fulfills one of the important requirements towards grids and that is scalability. The advantage of the proposal is the reuse of existing, stable architecture with a large installed base in today's Internet. Also the proposed light-weight architecture reduces the number of required components, compared to Globus Toolkit or HTCondor.

The future work aims at implementation of the improved architecture, presented in section 6. The improved architecture would support implementation of standard grid job scheduling schemes (SJF, FCFS, HRN). The request distribution time would be decreased with the improved architecture. There are open issues to be solved: fault tolerance, checkpoints, and job migration.

Acknowledgment

Authors would like to thank Marko Savic, Nikola Dragoljevic, Aleksandar Spasojevic, Marko Brett, and Aleksandar Stefanovic who took part in development of the prototype system, during their studies at the Chair of Computer Engineering, Faculty of Technical Sciences.

References

[1] (2006) Open Global Grid Forum. [Online]. https://www.ogf.org/documents/GFD.80.pdf

[2] Henning Schulzrinne, Gonzalo Camarillo, Alan Johnston, Jon Peterson, Robert Sparks, Mark Handley, Eve Schooler, Jonathan Rosenberg, "Session Initiation Protocol," in *RFC 3261.*: IETF, 2002.

[3] (2005, December) Open Global Grid Forum. [Online]. https://www.ogf.org/documents/GFD.136.pdf

[4] (2014, November) Website of MjSip: a complete javabased implementation of SIP (Session Initiation Protocol) stack. [Online]. www.mjsip.org

[5] Van Jacobson, Colin Perkins, Mark Handley, "SDP Session Description Protocol," in *RFC 4566.*: IETF, 2006, ch. 1.

[6] Rajiv Ranjan, Rajkumar Buyya, and Boualem Benatallah Mustafizur Rahman, "A taxonomy and survey on autonomic management of applications in grid computing environments," *Concurrency and Computation: Practice and Experience*, vol. 23, pp. 1990-2019, 2011.

[7] Henning Schulzrinne, Jonathan Lennox, "Transporting User Control Information in SIP REGISTER Payloads," in *RFC2026.*: IETF, 1999, ch. 10.

[8] F. Pollaczek, "Über eine Aufgabe der Wahrscheinlichkeitstheorie," *Mathematische Zeitschrift*, vol. 32, pp. 64–100.

[9] A. Y Khintchine, "Mathematical theory of a stationary queue," in *Matematicheskii Sbornik*, pp. 73–84.

[10] V. B. Iversen, *Teletraffic Engineering and Network Planning*. Denmark: Technical University of Denmark, 2011.

[11] C.D. Crommelin, "Delay probability formulas when the holding times are constant," *P.O. Electr. Engr. J.*, vol. 25, pp. 41–50.

[12] G.J. Franx, "A simple solution for the M/D/c waiting time distribution," *Operations Research Letters*, vol. 29, pp. 221-229, 2001.

[13] H.C. Tijms, "Stochastic Models, an Algorithmic Approach," in *Wiley*, New York, 1994.

[14] Jelena Misic, Vojislav Misic, Hamzeh Khazaei, "Modelling of Cloud Computing Centers Using M/G/m Queues," in *31st International Conference on Distributed Computing Systems Workshops*, Minneapolis, USA, 2011, pp. 87-92.

[15] Michal Hanuliak, Peter Hanuliak, "Performance modeling of parallel computers NOW and Grid," *American Journal of Networks and Communications*, vol. 2, no. 5, pp. 112-124, 2013.

[16] Jonathan Rosenberg, "Presence Event Package for the Session Initiation Protocol (SIP)," in *RFC 3856.*: IETF, 2004.

[17] Adam Roach, "SIP-specific event notification," in *RFC 3265.*: IETF, 2002.

[18] Regine Meunier, Hans Rohnert, Peter Sommerlad, Michael Stal, Frank Buschmann, *Pattern-Oriented Software Architecture Volume 1: A System of Patterns.*: Wiley, 1996.

[19] Vishnu Kant Soni, Manoj Kumar Mishra, Prachet Bhuyan Raksha Sharma, "A Survey of Job Scheduling and Resource Management in Grid Computing," *World Academy of Science, Engineering and Technology*, vol. 40, 2010.

[20] Ian Foster, "Globus Toolkit Version 4: Software for Service-Oriented Systems," *Journal of Comupter Science and Technology*, vol. 21, no. 4, pp. 513-520, 2006.

[21] Todd Tannenbaum, Miron Livny, Douglas Thain, "Distributed Computing in Practice: The Condor Experience," *Concurrency and Computation: Practice and Experience*, vol. 17, no. 2-4, pp. 323-356, 2005.

[22] Fang-Yie Leu, "A Wireless Grid Service Platform Using SIP and Agents," in *Proceedings of the Seventh ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, SNPD*, 2006, pp. 139-144.