A Network-Based Scalability Model for Distributed Real-Time Resource Management

Dominik Meiländer, Marcel Lorke, and Sergei Gorlatch

***Abstract*—We consider a challenging class of so-called Real-Time Online Interactive Applications (ROIA); popular examples include multi-player online computer games, e-learning and training based on real-time simulations. ROIA combine high demands on scalability and real-time user interactivity with the problem of an efficient and economic utilization of resources with heterogeneous network capabilities. This paper proposes a generic scalability model for ROIA that analyzes the application performance during runtime and predicts the demand for load balancing, i.e., whether and when to add/remove resources or redistribute workload. In an experimental evaluation, we prove the quality of our model by comparing the proposed workload distribution using our model against the optimal workload distribution for a multi-player online game.**

***Index Terms*—scalability model, network communication, resource management, Real-Time Online Applications (ROIA), Real-Time Framework (RTF).**

I. Introduction

THIS paper is motivated by the challenges of the emerging class of *Real-Time Online Interactive Applications (ROIA)*. Popular and market-relevant representatives of this application class are massively multi-player online games, as well as real-time training and e-learning based on high-performance simulation. ROIA are characterized by high performance requirements, such as: short response times to user inputs (about 0.1-1.5 s); frequent state updates (up to 50 Hz); large and frequently changing number of users in a single application instance (up to 10^4 simultaneously).

Since the high demands on ROIA performance usually cannot be satisfied by a single server, *scalability* (i.e., accommodating an increasing number of users) is achieved by employing distributed, multi-server application processing. Two main resources needed for the operation of ROIA are computational power and network bandwidth [1]. While we addressed the scalability of ROIA with respect to computational power in [2], this work focuses on scalability with respect to network bandwidth which is used for the communication between ROIA servers and their clients as well as for inter-server communication. Recently, ROIA providers increasingly use virtualized Cloud resources for ROIA, e.g., via Amazon EC2 [3]. The availability of potentially unlimited Cloud resources increases the demand for predicting application scalability and estimating the effect of different load-balancing actions, i.e., adding resources or redistributing workload.

In this paper, we focus on the scalability of ROIA with respect to network bandwidth when choosing among multiple load-balancing actions. Our contribution is a novel network-related scalability model that analyzes the performance of a ROIA application during runtime, predicts the maximum supported number of users and determines a suitable workload distribution for resources with heterogeneous network capabilities. We utilize our *Real-Time Framework (RTF)* [4] to provide a case study application that is used to evaluate the quality of the workload distribution proposed by our model.

The paper is organized as follows. Section II describes our target class of Real-Time Online Interactive Applications (ROIA) and the Real-Time Framework (RTF) used for their development and execution. Section III presents our new model for predicting the maximum supported number of users per server. Section IV reports our experimental results on the evaluation of the proposed workload distribution for an example multi-player online game. Section V compares our approach to related work and concludes the paper.

II. Scalable ROIA Development with RTF

Typically, there are three different actors involved in the development and execution of ROIA: (i) *Application developers* implement the ROIA application, i.e., server and client programs realizing the application logic, and suitable mechanisms for application state distribution and monitoring, (ii) *Application providers* make ROIA accessible to users by executing application servers on hardware resources and implement dynamic load balancing for ROIA sessions according to the current user workload, and (iii) *Users* connect their personal computers (*clients*) to application servers using the application client and control their avatars that interact with application *entities*, i.e., other users' avatars or computer-controlled characters, in the virtual environment. Each user accesses an application state which he shares with other users and interacts with them within one virtual environment.

We use the *real-time loop* model [5] for describing ROIA execution on hardware resources. Each user's client is connected to one application server that processes users' inputs (e.g., commands and interactions with other users), computes the application state updates and sends them to its users' clients (left-hand side of Fig. 1). One iteration of the real-time loop is called a *tick* and consists of three steps:

1) Each server receives inputs from its connected users.
2) Each server computes a new application state according to the application logic.
3) Each server sends the newly computed state to its connected users and to other servers.

Steps 1 and 3 involve communication to transmit the users' inputs and state updates between multiple application servers. The computation of a new application state (step 2) involves quite compute-intensive calculations which apply the application logic to the current state, taking into account the newly

Manuscript received August 25, 2014; revised September 01, 2014. Our research has received funding from the EC's 7th Framework Programme under grant agreements 318665 (OFERTIE) and 295222 (MONICA).

D. Meiländer, M. Lorke and S. Gorlatch are with the Department of Computer Science, University of Muenster, Germany e-mail: (see http://www.uni-muenster.de/PVS/en/mitarbeiter/).

ISBN: 978-988-19253-2-9
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

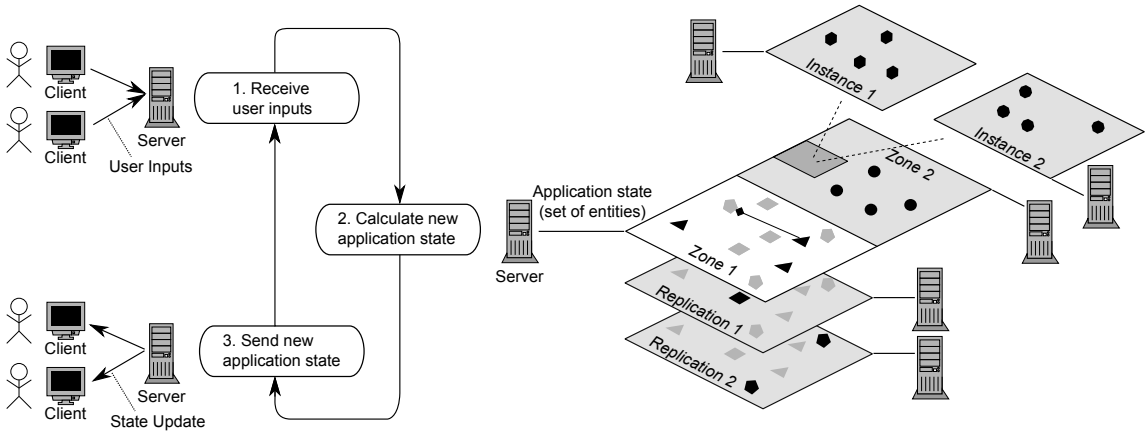

Fig. 1. One iteration of the real-time loop (left); RTF methods for application state distribution (right).

received users' inputs. The time required for one iteration of the real-time loop (*tick duration*) is directly related to the application's response time, and, hence, is used as a quality-of-experience criterion for ROIA.

The *Real-Time Framework (RTF)* [6] is our high-level development platform for ROIA which supports the application developer in three essential tasks as explained in the following: (i) application state distribution, (ii) communication handling, and (iii) monitoring and distribution handling.

RTF supports three common methods of *application state distribution* among servers (on the right-hand side of Fig. 1): zoning, instancing and replication, and combinations of them [6]. Zoning assigns the processing of the entities in disjoint areas (*zones*) to distinct servers. Instancing creates separate independent copies of a particular zone; each copy is processed by a different server. In the replication approach, each server keeps a complete copy of the application state, but each server is responsible for a disjoint subset of entities (*active entities*, black in Fig. 1) and receives updates for so-called *shadow entities* (grey in Fig. 1) from other servers.

RTF provides high-level mechanisms for *communication handling*: automatic (de-)serialization for objects to be transferred over network (user inputs, application state updates, etc.), (un-)marshalling of data types, and optimization of the bandwidth usage. RTF's *monitoring and distribution handling* allows the provider to change the distribution of the application state during runtime, as well as to receive monitoring data from RTF inside an application server. The distribution of the application state can be changed by load-balancing actions which include: (i) adding/removing resources to/from the application processing using zoning, instancing and replication, or (ii) migrating users between application servers, i.e., transferring the responsibility for user input processing and state update computation from one server to another.

III. The Scalability Model

In this section, we design a network-based scalability model for ROIA that will predict the *maximum number of clients* for each application server in order to predict when a server will become overloaded and to trigger some other load-balancing action, e.g., user migration from the overloaded server to the other servers. Obviously, the maximum possible number of clients of a zone depends on the number of servers assigned to it and the bandwidth of their network links.

Our model needs to solve an optimization problem, i.e., maximizing a *target function* given multiple input values (e.g., number of replicas) and non-trivial constraints expressed as inequations. For computing a solution to this optimization problem, we utilize the technique of *nonlinear programming* [7]. This allows us to model relevant constraints as a set of equations and inequations, and employ existing nonlinear programming tools, e.g., the CHOCO Solver Library [8].

Our model aims at maximizing the number of supported clients on N servers which is expressed by the following target function (a_i being the number of clients on server i):

$$n_{\max}(N) = \max\{\sum_{i=1}^{N} a_i \mid \text{given constraints are satisfied}\}$$

Our model is based on the *guaranteed bandwidth* of network links which denotes the bandwidth that is available to an application at any point in time. In contrast to the *available bandwidth* (i.e., the unused bandwidth) of network links, which is an often-used performance indicators for determining how many additional clients can be served [9], the guaranteed bandwidth is a stable value that is not influenced by other applications using the same links. For ROIA, it is common to specify QoS requirements for network links which makes it easy to determine practical values for the guaranteed bandwidth. Obviously, the guaranteed bandwidth is smaller or equal to the theoretically maximum bandwidth of the link and has the advantage that it allows for optimization without considering the current available bandwidth of any of the relevant links. In the following, we refer to the guaranteed bandwidth as bandwidth if not stated otherwise.

Our model addresses ROIA that are hosted in a datacenter. However, the underlying network topology that connects all servers has big influence on scalability issues. To take the network topology into account, we start with modeling a simple setup containing a single router in order to identify basic input values and constraints for our model (Section III-A). Then, we enhance our model for a more generic scenario involving multiple routers in a datacenter (Section III-B).

ISBN: 978-988-19253-2-9
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

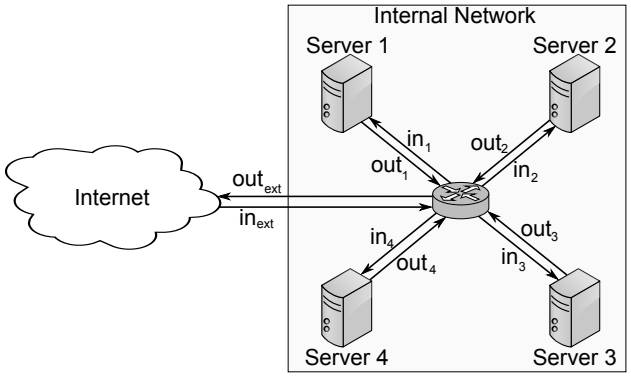

Fig. 2. Single router topology.

A. Single Router Topology

The *single router topology* is a very simple network topology that is actually very rarely used for ROIA in practice, but which allows us to determine basic input values and constraints for our model. Fig. 2 illustrates how several servers are connected to each other through a single router in a local-area network (LAN). The router is connected to an external network (e.g., the Internet) from which users connect to the application servers. We denote the bandwidth of the router's incoming and outgoing links to the external network by in_{ext} and out_{ext}, respectively. The bandwidth of the incoming and outgoing (internal) links of application server i are denoted by in_i and out_i, correspondingly, where $i < N$ and N is the number of application servers used for the replication of a particular zone ($N = 4$ in Fig. 2). Later on we will calculate approximating functions for in_{ext} (and other functions in our model) based on experimental results for a particular application.

In the following, we develop constraints for the single router topology. For this purpose we distinguish between (i) external network links, and (ii) internal network links.

The *external network links* receive client inputs as incoming traffic as illustrated in Section II (step 1 in Fig. 1). Hence, the total amount of incoming traffic from external network links is determined by the sum of bandwidth $input(a_i)$ required for receiving the inputs of a_i active entities from server i, where $i \leq N$ and N is the number of servers. Since the total amount of incoming traffic from external links is restricted by the bandwidth of the router's incoming link to the external network, our first constraint is expressed as follows:

$$\sum_{i=1}^{N} input(a_i) \;\leq\; in_{ext} \tag{1}$$

The outgoing external links are used for sending state updates to clients. The bandwidth $update(a_i,\ s_i)$ of server i to external network links is determined by the number of active entities a_i as well as by the number of shadow entities s_i: state updates are sent for each active entity and the size of state updates is determined by the overall number of clients (i.e., active + shadow entities; both kinds of entitites are explained in Section II) since each entity may change its state:

$$\sum_{i=1}^{N} update(a_i, s_i) \;\leq\; out_{ext} \tag{2}$$

The *internal network links* are used for communication between the application servers as well as between servers and clients. The latter is the part of the client-server communication within the internal network. The incoming links are utilized as follows:

- For a_i active entities of server i, bandwidth $input(a_i)$ (as used in (1)) is used for receiving input actions from the clients that are controlling these entities.
- For s_i shadow entities of server i, bandwidth $eu(s_i)$ is used for receiving entity updates from the servers that are responsible for the corresponding active entities. $eu()$ is a function that takes the number of entities for which updates are received (for incoming links) or sent (for outgoing links).
- For interactions of shadow entities with active entities of server i, bandwidth $fa(a_i,\ s_i,\ a_i + s_i)$ is used on server i for receiving forwarded actions from the servers that are responsible for these shadow entities. It is important to distinguish between the initiator and the target of an interaction between active and shadow entities since sending forwarded actions generates outgoing traffic on the initiator's server and incoming traffic on the target's server. Hence, $fa()$ is a function that takes the number of potential targets as first, the number of potential initiators of interactions as second, and the overall number of entities as third argument. For modeling incoming bandwidth on server i, the number of potential targets corresponds to a_i (active entities), the number of potential initiators corresponds to s_i (shadow entities), and the overall number of entities corresponds to $a_i + s_i$.

Since the overall incoming bandwidth of internal links used for input actions, entity updates and forwarded actions must be smaller or equal to the guaranteed bandwidth, we formulate the following constraint:

$$\underbrace{input(a_i)}_{\substack{\text{client-server}\\ \text{comm. within}\\ \text{internal network}}} + \underbrace{eu(s_i) + fa(a_i, s_i, a_i + s_i)}_{\text{inter-server comm.}} \leq in_i \tag{3}$$

The outgoing internal links are used for the following tasks:

- For a_i active and s_i shadow entities of server i, bandwidth $update(a_i, s_i)$ (as used in (2)) is used for sending state updates to the clients that are controlling the active entities.
- For a_i active entities of server i, bandwidth $eu(a_i)$ is used for sending entity updates to one other server in order to update its shadow entities. While there are $N-1$ other servers which require entity updates, we need to take into account the overhead caused by adding a new replication when one of the servers becomes overloaded. In this case, entity updates need to be sent to N other servers (including new replica).
- For interactions of active entities with shadow entities of server i, bandwidth $fa(s_i, a_i, a_i + s_i)$ is used on server i for sending forwarded actions to the servers that are responsible for these shadow entities. For modeling outgoing bandwidth on server i, the number of potential targets corresponds to s_i (shadow entities) and the

ISBN: 978-988-19253-2-9
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

number of potential initiators corresponds to a_i (active entities).

Since the overall outgoing bandwidth of internal links used for state updates, entity updates and forwarded actions must be smaller or equal to the guaranteed bandwidth, we formulate the following constraint:

$$\underbrace{update(a_i, s_i)}_{\substack{\text{client-server}\\ \text{comm. within}\\ \text{internal network}}} + \underbrace{N \cdot eu(a_i) + fa(s_i, a_i, a_i + s_i)}_{\text{inter-server comm.}} \leq out_i \tag{4}$$

With the previous information it is now possible to design a model that allows the calculation of the maximum number of clients per server. This is done by means of nonlinear programming. However, in order to do so, some additional constraints need to be added to the model. Since the complete application state is fully replicated at each server, the sum of shadow entities of any server must match the sum of active entities of all other servers, i.e.:

$$s_i - \sum_{n=1, n\neq i}^{N} a_n \;=\; 0 \tag{5}$$

It is then possible to calculate the maximum number of active and shadow entities $n_{\max}(N)$ for a given number of N servers. This is done by maximizing the number of active entities in the zone as shown in Equation (6):

$$n_{\max}(N) = \max\{\sum_{i=1}^{N} a_i \mid \text{constraints (1)-(5) are satisfied}\} \tag{6}$$

Since the number of shadow entities is related to the number of active entities, the former are indirectly maximized as well.

B. Datacenter Topology

The *datacenter topology* is a common network architecture for ROIA that are hosted in a single datacenter (Fig. 3). While the core router is still used to connect the datacenter and the external network, datacenters usually include multiple routers that can be distributed arbitrarily throughout the entire datacenter. This adds a substantial amount of new links to the model. we regard the internal connections in this case as *logical* rather than physical, since the servers are connected through a network of various numbers of routers and links, which is unknown. For this purpose, we denote the links that are used for communication between two servers $S1$ and $S2$ as the *path* between $S1$ and $S2$. A major challenge for this topology is that, even though the incoming or outgoing link to a server may have sufficient bandwidth, the path between two servers may not, due to some insufficient links on that path. In order to account for this, every path that connects a server to another server needs to be taken into account.

While the previous constraints (1)–(5) for the external and internal links must still hold true, we need to introduce additional constraints in order to address the utilization of logical links for communication (i) between the servers, and (ii) between the servers and the core router. Since it is possible that multiple paths between two servers exist,

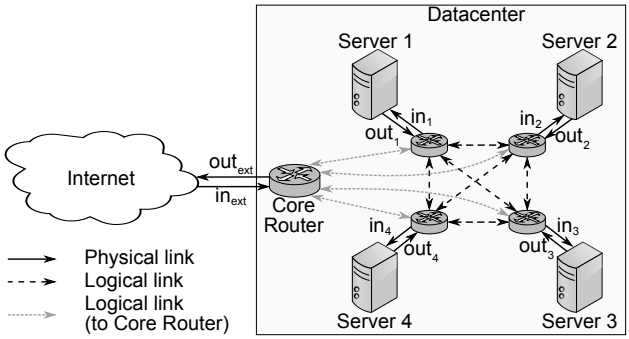

Fig. 3. Datacenter topology.

every single path has to be assessed when considering inter-server traffic. While one path between two servers may have sufficient bandwidth, the required amount of entity updates and forwarded actions can still not be exchanged if another path with insufficient bandwidth between these servers is chosen for communication. To account for this, we denote the link with the smallest bandwidth of all paths between two servers as *tight link* that determines the bandwidth for these paths. In order to identify the tight link for every pair of servers, we analyze all links that are part of paths between these servers. We denote the bandwidth between the servers i (sending) and j (receiving) as $path_{i,j}$. Let $link^k_{i,j}$ be the bandwidth of the k-th link between i and j and $num_{i,j}$ the total number of links between i and j, then the bandwidth between these servers is described as:

$$path_{i,j} \;=\; \min_{k=1,\ldots,num_{i,j}} link^k_{i,j} \tag{7}$$

For modeling additional constraints related to the inter-server communication, we have to consider entity updates and forwarded actions. The bandwidth $path_{i,j}$ must not be exceeded by the bandwidth required for (a) sending updates from server i to server j resulting in bandwidth $eu(a_i)$, and (b) sending forwarded actions initiated by the active entities of server i (denoted by a_i) to the targets managed by server j (denoted by a_j), resulting in bandwidth $fa(a_j,\ a_i,\ a_i + s_i)$. Since the outgoing traffic of server *i* is the incoming traffic of server *j* on the same server-server path, it is sufficient to focus on the outgoing traffic:

$$eu(a_i) + fa(a_j, a_i, a_i + s_i) \;\leq\; path_{i,j} \tag{8}$$

For modeling additional constraints related to the communication between the server and the core router, we have to consider client inputs and state updates. The bandwidth between server i and the core router, denoted as $in_{c,i}$ (incoming links) and $out_{i,c}$ (outgoing links), must not be exceeded by the bandwidth required for (1) receiving client inputs for a_i active entities on server i ($input(a_i)$) and (2) sending state updates for a_i active entities ($update(a_i,\ s_i)$):

$$input(a_i) \leq in_{c,i} \tag{9}$$

$$update(a_i, s_i) \leq out_{i,c} \tag{10}$$

For the datacenter topology, our enhanced model maximizes (6) for the given constraints (1)–(5) and (8)–(10).

ISBN: 978-988-19253-2-9
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

TABLE I
THEORETICAL MAXIMUM NUMBER OF ACTIVE AND SHADOW ENTITIES PER SERVER FOR UP TO 10 REPLICATION SERVERS.

# Servers	*# max. Clients*	*Active*	*Shadow*
1	385	385	0
2	543	272	271
3	665	222	443
4	766	192	574
5	855	171	684
6	939	156	780
7	1009	144	865
8	1080	135	945
9	1143	127	1016
10	1210	121	1089

IV. EVALUATION

As an application example for our experimental evaluation, we use the RTFDemo application which is a fast-paced action game from the domain of FPS (First-Person Shooters), with all typical features of modern online games. In RTFDemo, each user controls his own avatar (robot) in the 3D virtual world, and users can interact by shooting at (and thus damaging) other users avatars. For the sake of comfortable experiments with RTFDemo, users are simulated by so-called *bot clients* which are dedicated client processes that autonomously trigger movement and attack actions. Our experimental testbed uses the datacenter topology analyzed in Section III-B.

We conducted several benchmarking experiments for the determination of the different parameters of our model ($input$, $update$, etc.) and we used the *least squares approach* [10] to find functions that approximate our measurement data best. The least squares approach calculates the coefficients c_i of a given function $f(x) = \sum_{i=0}^{m} c_i \cdot x^i$, such that the resulting function optimally approximates the data points (x_i, y_i) from our measurements, i.e., $\sum_{i=0}^{m} (f(x_i) - y_i)^2$ becomes minimal. We chose polynomial functions of the second degree for approximation since most parameters are not growing linearly and thus using a polynomial of the first degree would lead to a large mean squared error (MSE). Furthermore, the average values have been used for the approximation of the data, so that the bandwidth consumption is not overestimated by outliers.

The nonlinear program resulting from our model yields the maximum number of active and shadow entities per server for an arbitrary number of servers. The results for up to 10 servers are shown in Table I, where the first column shows the number of servers replicating the zone, the second column contains the maximum number of clients, and the third and fourth column contain the maximum number of active and shadow entities on each server, correspondingly. The application servers are connected by 1 Gigabit network links.

In our evaluation, we compare the predictions for the maximum number of clients made using our model with the number of clients that is reached in practice before exceeding the network bandwidth. The bandwidth utilization is measured in four different setups, where in each setup the number of servers assigned to the zone is increased by one. During each of these measurements, the number of bots is increased at a constant rate of one bot every two seconds. The bots are distributed evenly between the servers at any point in time.

TABLE II
COMPARISON OF THE THEORETICAL AND THE OBSERVED MAXIMUM NUMBER OF CLIENTS.

# Servers	*theo. max. Clients*	*obs. max. Clients*	*max. Deviation*
1	385	380	1%
2	543	566	4%
3	665	696	4%
4	766	808	5%

The practical values are determined by measuring the bandwidth utilization for various numbers of servers. The client count at which one of the servers reaches its bandwidth limit is then compared to the results of Table I. For example, we observed that a single server is able to process 380 clients without violating performance requirements at any time. For 381 clients, we observed that performance requirements are violated infrequently due to the exceeded network bandwidth. However, since the behaviour of the application in that case is very application-specific we aim at avoiding performance violations at any cost.

The observed and the theoretical values are shown in Table II. The main reason for the observed deviation is arguably the unpredictable behaviour of bots which move and interact randomly. However, since the maximum deviation of the results is only 5% for a large number of clients, our model for the maximum number of clients and the use of nonlinear programming provides reasonable values for the estimation of the maximum number of clients.

V. RELATED WORK AND CONCLUSION

This paper develops a novel analytical model for Real-Time Online Interactive Applications (ROIA) that analyzes the application performance during runtime and predicts the demand for load-balancing actions on distributed platforms with multiple servers, like Grids and Clouds. We have shown how our scalability model addresses network topologies practically used for ROIA processing in datacenters. In a practical case study of a dynamic multi-player online game, we demonstrated the practical relevance of our scalability model by showing that the deviation between the proposed workload distribution and the optimal distribution is less than 5%.

There are other parallelization and load-balancing concepts that aim at realizing a large number of users in a ROIA. BHARAMBE2006 et al. propose that a server responsible for an entity tries to predict which entities from other servers it needs in order to serve its own clients, and only requests these. To do so it uses the *Area of Interest* (AoI) of its entities and a *distributed hash table* (DHT) shared across the servers. By storing range-queries that describe the AoI of an entity, servers can perform look-ups and predict the probability of an interaction [11]. This is fundamentally different from the RTF approach, where all entities are replicated to all servers at all times. Another concept described in [12] and [13] uses so-called *microcells* to distribute the load between the servers. In this approach, a zone (or the virtual environment as a whole) is divided into microcells, which can be distributed arbitrarily to any server. Should a server reach its performance limit, it migrates the clients of one

ISBN: 978-988-19253-2-9
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

of its microcells to another server with sufficient bandwidth. Approaches based on microcells are particularly suitable for clustered user distributions, i.e., when few areas of the virtual environment have very high user density while large parts of the environment remain unattended by the users. The replication approach targeted by our scalability model divides the virtual environment into zones which are typically larger areas, and it allows providers to use multiple application servers for the computation of a single zone via its replication. Our approach is arguably more suitable for uniform user distributions. While both approaches involve overhead for inter-server communication, our scalability model enhances the usability of the replication approach.

The first mathematical models for computer networks developed in the 1960's make use of *queueing theory* to model the arrival rate of packets at network nodes using stochastic processes; packets are then distributed among the outgoing links of the nodes [14]. However, the cost of acquiring new bandwidth is not growing linearly and the network topology is often enormous, making it unfeasible to only consider arrival rates. Therefore, models were extended towards the use of nonlinear programming, with the aim of minimizing the costs while providing sufficient bandwidth for each link of the network [15].

KELLY et al. proposed a *Utility* function which represents the utility of a single user of the network [16], in order to allocate the available network resources in a fair, distributed and stable manner among the clients. This is also known as *network utility maximzation* (NUM) [17]. An extension of the NUM approach is described in [18], where the original master problem is decomposed into several smaller subproblems which can then be solved on their own. To do so, the master problem allocates a fraction of the existing resources to each subproblem. In [19] the modeling of computer networks has been separated into a *macroscopic* and a *microscopic* view. The macroscopic view is also referred to as *static* routing and aims at an efficient resource usage of the network as well as an optimal flow of the data. An example of this level of granularity are the *Autonomous Systems* of the Internet. The microscopic view is also referred to as *dynamic* routing and optimizes TCP windows sizes as well as the routing of each individual packet at each hop.

More recently, these approaches have been extended from a best-effort perspective to one that considers QoS requirements of different kinds of applications. For example, the tolerance of sending an e-mail is much higher with regards to delays and packet losses than real-time applications such as ROIA or video-on-demand (VoD) services [20]. RAJKUMAR et al. therefore propose a *QoS-based Resource Allocation Model* (Q-RAM) that satisfies multiple requirements of these applications, such as timeliness, reliable packet delivery and simultaneous resource access [21]. These problems, which are also relevant in the context of this paper, were analyzed in [22] for a VoD service which also needs to ensure QoS for the satisfaction of its customers. Nonlinear programming is used there to obtain an optimal bandwidth reservation that prevents under-provisioning, but also minimizes the over-provisioning of system resources.

REFERENCES

[1] S. Gorlatch, F. Glinka, and A. Ploss, "Towards a Scalable Real-Time Cyberinfrastructure for Online Computer Games," in *Proceedings of the 15th International Conference on Parallel and Distributed Systems*. Shenzhen, China: IEEE Computer Society, Dec. 2009, pp. 722–727, iSBN 978-0-7695-3900-3.

[2] D. Meiländer, S. Köttinger, and S. Gorlatch, "A Scalability Model for Distributed Resource Management in Real-Time Online Applications," in *42nd International Conference on Parallel Processing (ICPP)*. IEEE, 2013, pp. 763–772.

[3] "Amazon Web Services (AWS)," http://aws.amazon.com/game-hosting, 2014.

[4] "Real-Time Framework (RTF)," http://www.real-time-framework.com, 2014.

[5] L. Valente, A. Conci, and B. Feijó, "Real Time Game Loop Models for Single-Player Computer Games," in *SBGames '05 – IV Brazilian Symposium on Computer Games and Digital Entertertainment*, 2005.

[6] F. Glinka, A. Ploss, S. Gorlatch, and J. Müller-Iden, "High-Level Development of Multiserver Online Games," *International Journal of Computer Games Technology*, vol. 2008, no. 5, pp. 1–16, 2008.

[7] D. Luenberger and Y. Ye, *Linear and Nonlinear Programming*, ser. International Series in Operations Research & Management Science. Springer, 2008.

[8] "CHOCO Solver Library," http://www.blaast.com, 2014.

[9] R. Prasad, C. Dovrolis, M. Murray, and K. Claffy, "Bandwidth Estimation: Metrics, Measurement Techniques, and Tools," *IEEE Network*, vol. 17, no. 6, pp. 27–35, 2003.

[10] C. R. Rao, H. Toutenburg, C. Heumann, T. Nittner, and S. Scheid, *Linear Models: Least Squares and Alternatives*, 1999.

[11] A. Bharambe, J. Pang, and S. Seshan, "Colyseus: A Distributed Architecture for Online Multiplayer Games," in *NSDI'06 Proceedings of the 3rd conference on Networked Systems Design & Implementation - Volume 3*, 2006, pp. 155–168.

[12] B. De Vleeschauwer, B. Van Den Bossche, T. Verdickt, F. De Turck, B. Dhoedt, and P. Demeester, "Dynamic Microcell Assignment for Massively Multiplayer Online Gaming," in *NetGames '05 Proceedings of 4th ACM SIGCOMM workshop on Network and system support for games*, 2005, pp. 1–7.

[13] J. Chen, M. Delap, and H. Lu, "Locality Aware Dynamic Load Management for Massively Multiplayer Games," in *PPoPP '05 Proceedings of the tenth ACM SIGPLAN symposium on Principles and practice of parallel programming*, 2005, pp. 289–300.

[14] L. Kleinrock, *Communication Nets: Stochastic Message Flow and Delay*, 1972.

[15] ——, "On the Modeling and Analysis of Computer Networks," *Proceedings of the IEEE*, vol. 81, no. 8, pp. 1179–1191, 1993.

[16] F. Kelly, A. Maulloo, and D. Tan, "Rate control for communication networks: shadow prices, proportional fairness and stability," *Journal of the Operational Research Society*, vol. 49, no. 3, pp. 237–252, 1998.

[17] R. Srikant and L. Ying, *Communication Networks*, 2011.

[18] D. P. Palomar and M. Chiang, "A Tutorial on Decomposition Methods for Network Utility Maximization," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 8, pp. 1439–1451, 2006.

[19] A. Ephremides and S. Verdu, "Control and Optimization Methods in Communication Network Problems," *IEEE Transactions on Automatic Control*, vol. 34, no. 9, pp. 930–942, 1989.

[20] S. Shenker, "Fundamental Design Issues for the Future Internet," *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 7, pp. 1176–1188, 1995.

[21] R. Rajkumar, C. Lee, J. Lehoczky, and D. Siewiorek, "A Resource Allocation Model for QoS Management," in *RTSS '97 Proceedings of the 18th IEEE Real-Time Systems Symposium*, 1997, pp. 298–307.

[22] D. Niu, H. Xu, B. Li, and S. Zhao, "Quality-Assured Cloud Bandwidth Auto-Scaling for Video-on-Demand Applications," in *Proceedings of the IEEE INFOCOM 2012*, 2012, pp. 460–468.

ISBN: 978-988-19253-2-9
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)