

Badumna Network Suite: A Decentralized Network Engine for Massively Multiplayer Online Applications

Santosh Kulkarni

NICTA, Victoria Research Laboratories
santosh.kulkarni@nicta.com.au

Abstract

We present Badumna Network Suite, a network engine for Massively Multiplayer Online (MMO) applications. MMO applications such as World of Warcraft and Second Life use client-server architecture. This architecture has several drawbacks such as high deployment costs, single point of failure and lack of scalability. Badumna's goal is to scale to truly massive player counts using minimal operator owned infrastructure and network resources. The key to achieving this goal is in forming a peer-to-peer network and distributing the processing on this network. By doing this, Badumna reduces hosting costs significantly and it also increases the maximum number of users that can be allowed in a given region. The technology comprises of a distributed network engine that interfaces with existing MMO platforms. This paper presents the technology and discusses the issues involved in integrating such a technology with commercial gaming platforms – the expectations and the challenges. We then present results from commercial trials and conclude by discussing the role peer-to-peer computing will play in defining the future of MMO technology.

1. Introduction

Massively Multiplayer Online (MMO) applications such as World of Warcraft and Second Life are gaining immense popularity in recent years. These applications allow thousands of users from all around the world to interact with each other simultaneously in a persistent environment that is built using 2D or 3D content. These applications use a server-based architecture to manage the application functionality such as synchronization of real-time data, object updates, etc. This poses several inherent problems. Servers can handle a fixed number of users. The infrastructure cost to support a large number of users is very high as it

involves managing massive server farms. Handling flash crowds is also a big problem as servers are unable to handle a sudden increase in the number of peak users. A centralized architecture is also prone to single point of failure.

Decentralized architectures that are based on the use of peer-to-peer networks are emerging as an interesting alternative to support such applications [1, 2]. In this paper, we present Badumna, a decentralized engine that provides a complete networking framework for MMO applications. Badumna's goal is to provide a network engine that is highly scalable, robust, secure and reliable. The key to achieving this objective is in forming a structured peer-to-peer network of all the users and distributing the load of the server across the end-users. Badumna is available as a network engine for MMO applications and it has been integrated with several commercial MMO platforms. This paper briefly presents the technology and focuses on the issues that arose from the commercial integration process. Results from the commercial trials are presented followed by a summary that discusses the role peer-to-peer networks will play in defining the future of MMO technology.

2. Badumna Network Suite

Badumna network engine provides a complete networking framework for implementing MMO applications. Figure 1 shows the logical components that are part of Badumna. A brief description of the components follows.

Network façade: The network façade is the primary interface to access Badumna's functionality. It consists of an API [5] that can be used by application programmers to call the various functions that are supported by Badumna. The primary aim is to provide a simple and flexible set of operations to the application developers and hide all the complex functionality from them.

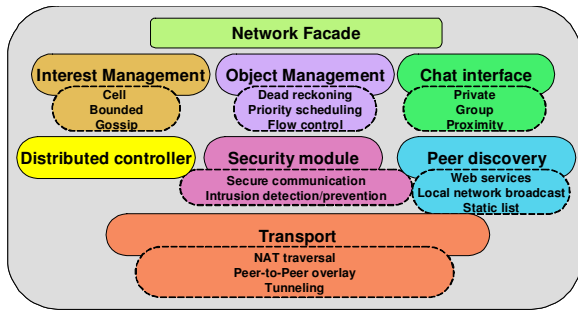


Fig. 1 Badumna – logical components

Interest Management: The interest management (IM) service is responsible for providing relevant real-time information to all the entities in the MMO application and ensuring their game state is accurate and updated. In the absence of a centralized indexing server, Badumna relies on distributed spatial indexing algorithms that can be executed on the peer-to-peer network. Badumna’s IM service treats all dynamic objects within the virtual environment as entities. This includes all the avatars (users), non-playing characters such as monsters and dynamic content such as a light-switch and moving cars.

Badumna’s aim is to provide an IM service that is scalable so that it can handle flash crowds and at the same time is also reliable and accurate. Badumna’s IM service consists of a tiered approach that switches between three different protocols depending on the network conditions in the virtual environment. The three IM protocols are cell, dynamic bounded, and gossip. Cell protocol divides the entire MMO space into fixed sized regions (cells) and inserts these cells in the peer-to-peer network that is formed using a DHT. This ensures that the load is adequately balanced. Peers closest to the cell assume responsibility for that cell (cell server). When an entity first joins the application, it uses the cell protocol and inserts itself and its interest region into relevant cell servers. One of the drawbacks of using the cell protocol is in the large overhead involved in maintaining the DHT. Every single insert in the DHT is replicated four times. Therefore, if there are a large number of entities present within a region, they would introduce a significant amount of traffic in the network.

Dynamic bounded protocol addresses this issue by reducing the load on the DHT thereby ensuring that it remains stable even when a large number of entities are in the virtual environment. A dynamic bounded region is a region of space that can be of any shape and size. Unlike a cell, a bounded region is dynamic – it can

move around in space and resize itself as required. Furthermore, a bounded region is an entity, so it can be inserted into other IM services (e.g. cells or other bounded regions). A bounded region can take on any shape supported by the underlying object system. Dynamic bounded regions introduce an additional level in the discovery process. Each entity and its associated interest region are managed by a bounded IM region. The bounded region queries the DHT to find out about other entities and other bounded regions of interest. The entities and the interest region query their bounded region to identify the remote objects of interest. As there is only one DHT request for each bounded region (which may contain multiple entities), this protocol significantly reduces the traffic on the DHT. Figure 2 shows an example of a dynamic bounded region. Bounded regions reduce the load on the DHT and hence minimize the subsequent overheads attached with replication. Bounded regions have a limit in terms of the number of entities they can support. Since bounded regions are controlled by individual peers they could get overloaded if a large number of entities joined the region. This situation could occur if a large number of users clustered in a small space. It would result in overloading the responsible peer and make the system unstable.

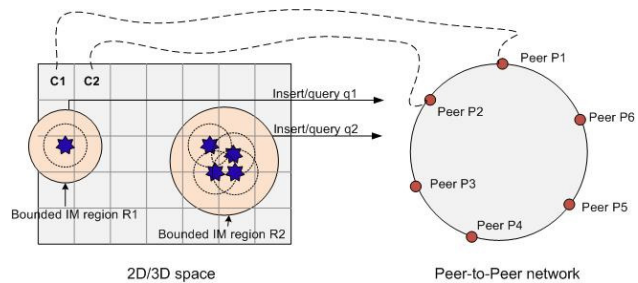


Fig. 2 Dynamic bounded IM region

Gossip protocol ensures that peers don’t get overloaded no matter how many entities enter a region or a cell. An entity enters gossip mode when its number of intersections exceeds a certain threshold. The number of intersections is proportional to the total number of objects in the region. Gossip protocol ensures that the IM service can scale to a large number of users without degrading the performance of the overall system. It does that by restricting the number of queries to the cell server to a threshold that is fixed by the system depending on the cell server’s capacity. Depending on certain parameters it reduces the frequency an entity should interact with the cell server. Entities rely on neighboring entities to inform each

other about other entities in the region. Each entity always inserts itself into the DHT (using the cell protocol) the first time it enters the scene. However, as the number of intersections for the entity exceeds the predetermined threshold, it starts gossiping with its neighboring entities. While an entity is in gossip mode, it will still query the cell server periodically at a reduced frequency. This ensures that the cell server has knowledge of the entities and can rectify any discrepancies. The cell server query frequency for an entity is computed using a probability function that takes into account the number of intersections for that entity.

Badumna's IM service dynamically switches between the three protocols depending on various parameters such as the nature of the MMO application, size of the 2D/3D space, and region density.

Object management: This component includes the interaction manager and is responsible for handling object updates. The service provides a mechanism for the application to send reliable and unreliable updates via Badumna. It also provides an extremely flexible framework allowing application programmers to define customized data types and send updates using those data types. Badumna aims to provide a solution that enables highly responsive interaction at each peer and ensure that all the users have a smooth experience. It makes use of techniques such as dead reckoning (send updates only when there is a state change), priority scheduling (prioritize updates and dynamically change update frequency based on various parameters), and flow control (modify update frequency depending on remote peers capacity) to achieve its objective.

Chat interface: The chat interface on Badumna provides a means to exchange text messages between peers using Badumna's network. Badumna supports three different types of chat messages: private chat (secure chat interface between two entities), group chat (secure chat between a group of entities within a scene or across multiple scenes), and proximity chat (secure interface to send messages to all entities within an entities' visible region).

Distributed controller: The distributed controller provides an ability to execute code in the network and make sure that it is only running on one machine at any one time. Using the concepts of distributed agents, it ensures that MMO game logic is executed in the network in a reliable and consistent fashion and MMO developers do not have to worry about the complexities of network management. Distributed controllers can

run over any network (client, server, combination). Distributed controllers are useful to support persistence and dynamic objects within a scene such as NPCs.

Security framework: Security is provided for different operations within Badumna. The level of security depends on the nature of the operation. MMO developers have full control over the level of security they can provide for the different operations. The framework supports secure authentication (peers exchange certificates before forming connections), encryption (all communication is encrypted using shared keys), and intrusion detection (responsible for detecting malicious activity and monitoring the network).

Peer discovery: This module is responsible for discovering other peers in the network when a new peer joins the network. Three different approaches are used to achieve this objective. Web service approach publishes a list of last N peers that joined the network. This information is used by new peers to discover a small number of initial peers and join the network. Local network broadcast sends a message across the local network to discover any peers that are part of the Badumna network. This approach ensures that peers that are part of a local network can find each other irrespective of the network being behind a NAT device. Certain MMO applications use a set of peers that are operator controlled and are introduced into the network before any other peers are allowed to join.

Transport: The transport layer is responsible for all the underlying communication between the different peers. It provides mechanisms to support TCP and UDP packets. Object updates are sent as UDP packets. The transport layer provides a facility to transmit reliable and unreliable UDP packets. Several mechanisms are in place in the transport layer to support NAT devices. Majority of the NAT types (symmetric, asymmetric, restricted port, etc) have been catered for in Badumna. With most of the NAT devices, all the traffic is sent directly between the peers without the need for relay. If two peers on NAT devices are unable to communicate directly, then tunneling is used in such a case. Tunneling is also used if a peer is behind a firewall that blocks UDP traffic. An http tunnel is established between the blocked peer and a tunneling peer. The tunneling peer is responsible for all the communication and handles all the object updates, inserts/retrievals from the interest management service for the blocked peer.

3. Commercial integration

Integrating Badumna with commercial MMO applications was a challenging task that involved overcoming several issues and expectations from the commercial partners. Commercial MMO operators are used to the level of service and support provided by client-server network engines. This includes complete network monitoring, transaction logging, server validation, and support for secure transactions. Operators want complete knowledge of their MMO application in terms of the number of users in the system at any time and the type of operations performed by the users. Operators also expect a certain level of security that is comparable to what they receive with current MMO architectures. In a peer-to-peer MMO architecture every client acts as a potential server and may be responsible for a subset of the game state information. If the client software is compromised, malicious users can try and disrupt the operation of the application as they may be able to manipulate game state information. Operators are paranoid about malicious users trying to break into their MMO application and disrupting the service. MMO operators also expect the user experience to be comparable to the level achieved with client-server architectures. Operators also want the network engine to be integrated with their MMO platform without major changes to their platform and have the ability to switch between peer-to-peer and client-server modes.

Badumna uses two different integration approaches – active and passive. The active integration approach involves linking Badumna DLLs within the MMO technology development platform. This provides the MMO application developers complete access to Badumna's APIs. MMO developers can therefore choose whether they want a specific operation to use the client-server network engine or the peer-to-peer network engine. The entire application is also executed as a single process making it an optimized solution. This approach is used when the MMO technology platform is compatible to that of Badumna and the linking process is fairly straightforward. Active integration process was used with VastPark [3], a global provider of virtual worlds' platform. The passive approach involves running Badumna as a separate process on each client machine. This approach is used if the technology platforms are not fully compatible and the operator does not want to modify their MMO platform. In passive integration, Badumna runs as a proxy, intercepting all traffic to and from the MMO application. All the traffic that can be managed

by Badumna is sent over its peer-to-peer network and the rest of the traffic is sent to the MMO server. This integration process was used with OpenSim [4], an open-source virtual worlds' platform.

Badumna also provides a control center that maintains global state information. All the entities send a heart-beat message periodically to the control center. This enables the control center to keep track of the total number of users in the network. It also allows the control center to monitor the network and make sure that the MMO application is operating normally. The overheads introduced by the control center are minimal and give the operators confidence in using Badumna as they are able to visualize the entire network. Badumna's security is very critical for the operators as it gives them the assurance that Badumna can detect malicious activity and remove responsible users from the network.

4. Commercial trials

Badumna is currently distributed with VastPark's virtual worlds' platform. Prior to releasing the product, a number of trials were conducted with VastPark's beta testers from around the world. Fig. 3 shows the results from one of the public trials. This trial had 15 users from around the world enter a virtual room and interact with each other. The graph shows the total traffic (inbound and outbound) at the server. It can be observed that the traffic is negligible throughout the test. The peaks in the graph represent a new user joining the network as they need to download the content to their PC.

We also conducted a set of tests with the OpenSim platform. Since we had a passive integration with OpenSim, we were able to measure the traffic by turning Badumna on and off. Figure 4 shows the traffic (inbound and outbound) at the OpenSim server when Badumna is disabled and Figure 5 shows the traffic when Badumna is enabled. It can be clearly seen that Badumna reduces the server traffic significantly (up to 75%). The traffic per peer used by Badumna is capped at 64Kbps and the Badumna process uses around 5% of the CPU on a standard dual core laptop.

The results show that by introducing a small processing and network overhead at each peer, Badumna is able to reduce the server requirements dramatically and provide an MMO solution that is cheaper, scalable, reliable and secure.

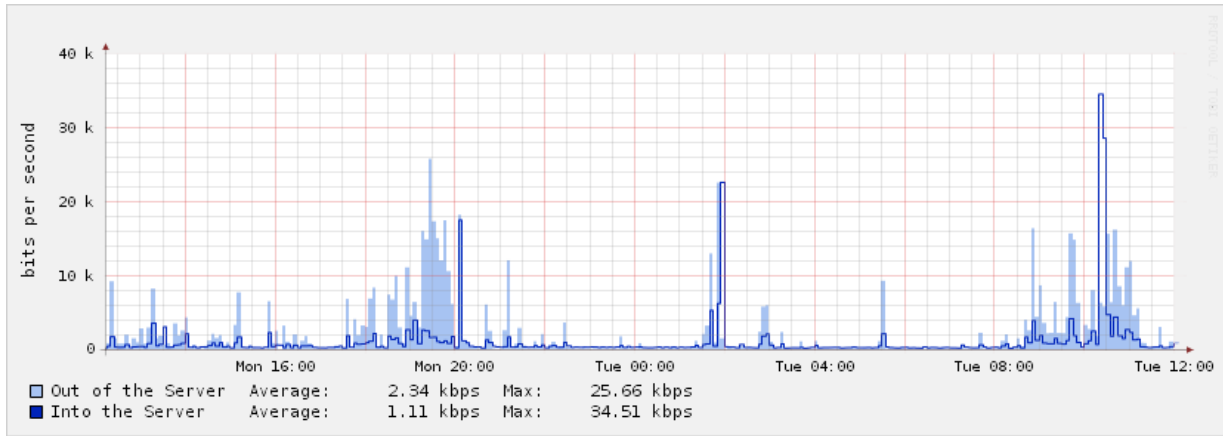
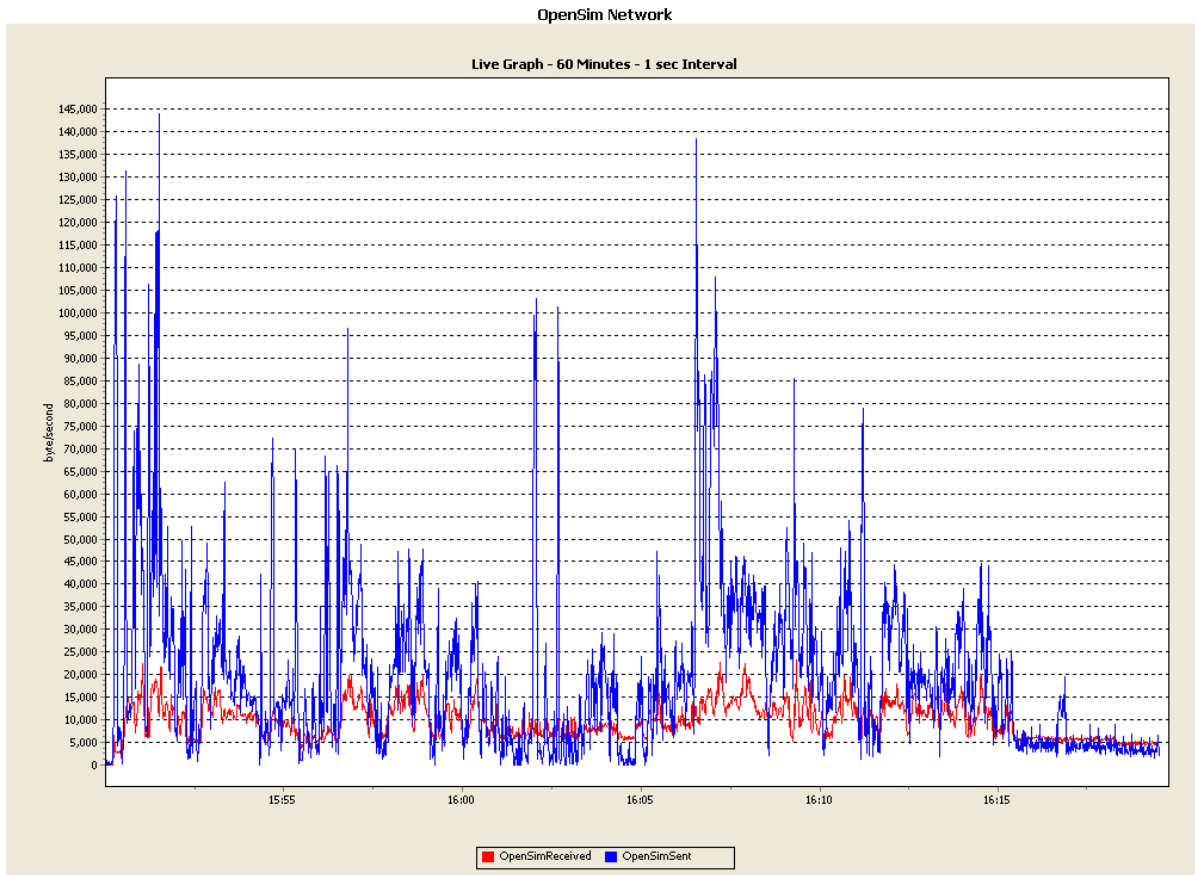


Fig. 3 Server load – VastPark trial



PRTG Traffic Grapher V6.2.1.962 - 20/02/2009 4:24:02 PM

Fig. 4 OpenSim traffic without Badumna

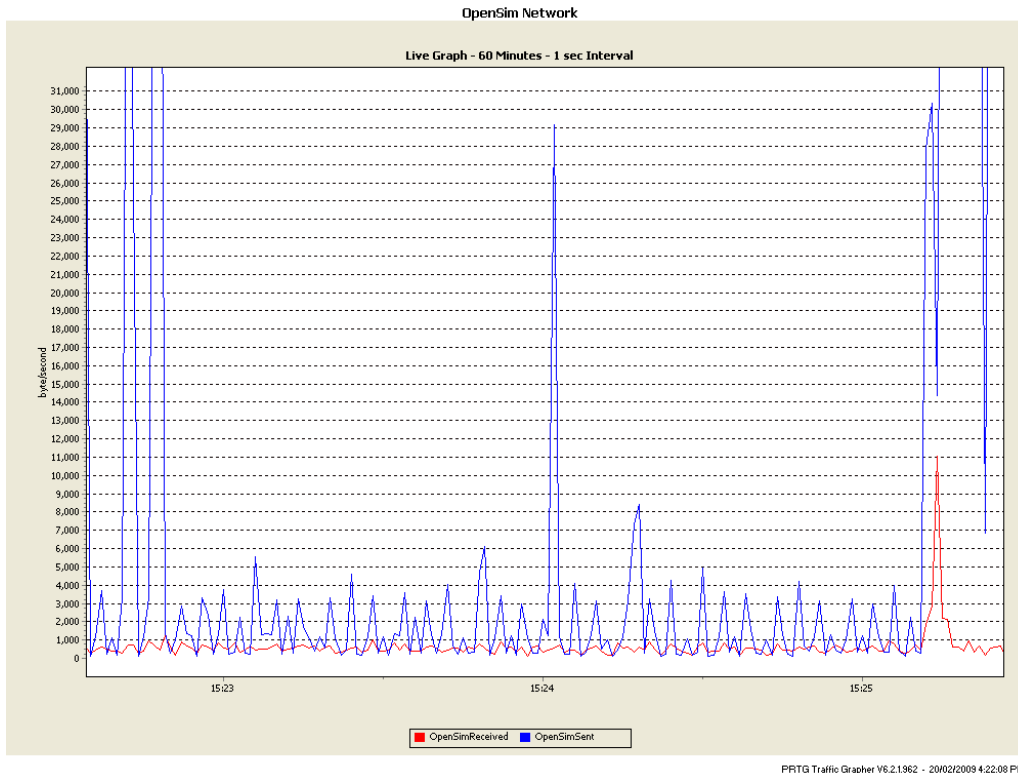


Fig. 5 OpenSim traffic with Badumna

5. Summary

Peer-to-peer networks have the potential to significantly improve the scalability of MMO applications. MMO operators are able to appreciate the value that peer-to-peer networks have to offer. However, the practical issues with peer-to-peer networks such as unreliability, heterogeneity, and untrustworthiness make it harder for commercial operators to accept a pure peer-to-peer network architecture.

In the short-term, hybrid MMO architectures that utilize a peer-to-peer and client-server architecture will start to emerge as new MMO applications are deployed. These architectures will have a set of operations such as chat and non-critical updates that are supported using a peer-to-peer network and the rest of the operations that require validation are supported by the server. Studies have shown that in a typical MMO application, 80% of the traffic is non-critical (such as chat and position updates) and only 20% of MMO traffic is critical (that requires server validation). Commercial operators are therefore comfortable with such architectures as they are able to exploit the

benefits of using a peer-to-peer network while still not compromising on the security and reliability that is offered by the servers. As further research progress is made in the field of trust management and security models for peer-to-peer networks, we will see more and more MMO operations being supported on the peer-to-peer network. Industry engagement is critical as it allows researchers to learn about the requirements and problems; and obtain valuable feedback from the end users of the technology. In the long-term, peer-to-peer network architectures have the potential to become the default standard for MMO applications.

6. References

- [1] S. Hu, J. Chen, and T. Chen, VON: a scalable peer-to-peer network for virtual environments, *IEEE Network*, vol 20, issue 4 (2006), 22-31.
- [2] J. Keller, and G. Simon, Towards a peer-to-peer shared virtual reality, in *Proc. 22nd International Conference on Distributed Computing Systems*, July 2002, 695-700.
- [3] VastPark, <http://www.vastpark.com>
- [4] OpenSim, <http://www.opensim.org>
- [5] Badumna Network Suite, <http://www.badumna.com>