TECHNICAL
SOLUTION
GUIDE

DIGITAL MEDIA

Digital Media: Massively Multiplayer Online Gaming (MMOG)

Game developers call the art of optimizing code for a particular game platform "getting close to the metal." Butterfly.net has created a powerful architecture for online game developers who are looking to optimize game performance through tight integration of hardware and software—that is, get close to the metal—in a high-performance way.

The company believes massively multiplayer (MMP) online gaming using symmetric multi-processor (SMP) computing has significant value within an online gaming development, production, and delivery environment.

Solution Provider
The Butterfly.net Grid

Solution Focus Area
Subscription Media Services—MMOG

Solution Architecture
Massively Multiplayer Online Gaming Grid

> "The video game market is expected to grow from $10 billion in 2001 to $18 billion in 2005[1], and that's just in the United States."

Defining Online Gaming and Massively Multiplayer Games Technologies

Computer games today are incredibly complex. With each new generation of hardware, developers can create even more fantastic universes in which players can immerse themselves. As the universes in games grow more complex—more worlds, more buildings, more rooms, with more polygons, textures, characters, and so on—so does the development, distribution, and infrastructure needed to support them.

The video game market is expected to grow from $10 billion in 2001 to $18 billion in 2005[1], and that's just in the United States. A trend of this magnitude will experience growing pains, and online gaming is no exception. The Internet has brought to the gamer what standalone systems never could: millions of users on a packet network investing enormous amounts of time, energy, and money in the pursuit of adventure, status, power, fellowship, and fun in persistent-state, massively multiplayer games (MMGs).

Examples such as EverQuest*, the first 3D-hardware-required MMG, claims a half-million subscribers. NCSoft estimates that each month 1.2 million people pay to play Lineage[2], its South Korean MMG. But as engrossing and compelling as current MMGs are for some players, they leave most gamers cold, for several reasons:

- MMGs are often unavailable for hours at a time. By contrast, standalone games offer rapid response times.
- MMGs suffer from lag. Standalone games host intelligent adversaries and allies in thought-provoking conflicts.
- MMGs offer a minimal set of interactions.

One reason that MMGs do not currently operate as well as standalone games is that MMG developers do not have a development platform on which to write and test their games. For example, standalone game developers, writing for personal computers, can optimize their code for DirectX*. Developers creating games for console devices, such as the Sony* PlayStation* 2 (PS2), can write to the Emotion Engine*. In these cases, developers have the opportunity to "code to the metal" of a particular device or platform and the peripherals that each platform supports—to optimize game performance through tight integration of hardware and software.

MMGs also suffered from conventional network infrastructure problems that have developed as quickly as the number of online gamers. Legacy servers based on inflexible, monolithic architectures are the source of the problems. To overcome server limitations, game designers must divide games into "shards" that provide copies of each game world on separate servers. Generally, each shard supports a maximum of around 4,000 gamers. As a result, MMG developers struggle with complex network and software balance issues, which distracts them from developing the best games possible.

Game publishers, in turn, must manage and support homegrown technologies for each game. This limits their ability to build effective, repeatable, and reliable infrastructures that support multiple MMGs and titles. They are forced to a high price for poor reliability and support costs, even as valuable revenue diminishes, thanks to server maintenance and reconfiguration that shuts down or slows down entire games.

[1] *Intel® Business Center Case Study*, "Butterfly.net Uses Intel® Architecture to Build a Global Gaming Grid", Pg.1, Copyright ©2003, Intel Corporation

[2] *The Butterfly Grid: Technical Architecture Overview*, Pg.7, Copyright 2002 Butterfly.net, Inc.

Performance and reliability issues severely impact an online gamer's experience and compromise their ability to interact with online friends. As the number of gamers per shard increases, so does latency within traditional MMGs. To avoid long delays to attach to popular game servers or the need to log in and out of MMG levels, developers must restrict the size of game worlds per server—which limits the players' experience and satisfaction level in another way.

To allow online games to evolve into a profitable and mature market, a cost-effective, scalable, high-performance, and reliable infrastructure is desperately needed.

Butterfly.net, faced with the architectural challenge that the MMG market presented, turned to Intel® Architecture and grid computing.

Balancing Architecture Decisions

As early as July 2000, Butterfly.net began to study the problems associated with developing and deploying online games. Grid computing—an emerging technology already used successfully by the scientific and financial communities—seemed promising. Grid computing is essentially a network of servers acting as one supercomputer for intensive distributed data processing. Butterfly.net thought that grid computing technology applied over the Internet could be used to develop an architecture and approach for using the Internet as a high-performance game machine.

"To allow online games to evolve into a profitable and mature market, a cost-effective, scalable, high-performance, and reliable infrastructure is desperately needed."

Grid computing has the potential to revolutionize the way online games are developed, distributed, and played. If a server goes offline—as in distributed data processing—the grid's self-healing properties seamlessly route game play to the nearest available server, resulting in a more satisfying experience for gamers.

The team at Butterfly.net realized that to be as efficient and reliable as other broadcasting industries, they would have to architect a distributed, multicast-mesh over user datagram protocol (UDP) that provided load-balanced and distributed computing globally—while maintaining a competitive cost structure for the MMG community. Given the complexity of the project, Butterfly.net also realized that whatever platform they ultimately selected, it would have to support a development and production model that used standard protocols, standard products, and open source technologies.

These factors ultimately resulted in three key architecture choices:

- Intel Xeon™ processor-based systems
- Blade server technology
- Linux* as the core operating system

Intel Xeon processors enable the higher transaction rates and fast response times needed to optimally serve high transaction volume messaging and complex, highly graphical, responsive gaming applications. Use of high-performance Intel Architecture ensures that the servers have the power, performance, and reliability necessary for such applications. In addition, Intel Xeon processors support a broad choice of interoperable, flexible hardware and software building blocks to customize and optimize game development and server infrastructure. The scalability of the hardware platform ensures a long-term upgrade path and capacity for expansion.

Blade server technology is a high-density computing power packaging technology that is well suited to Web serving, caching, load balancing, streaming media, and firewall protection.

Linux is a UNIX* work-alike operating system created by Linus Torvalds and a motivated, active group of programmers who united via the Internet. Thanks to years of source code fine-tuning by this group of dedicated volunteers, Linux has achieved enterprise-class reliability, ease of administration, and outstanding performance. Linux delivers more speed and power from the hardware (especially Intel Architecture, on which it was originally developed), provides better security than many commercial alternatives, and is very reliable.

Many developers in the online gaming community have switched to application development on Linux-based systems. Because the Butterfly.net system was designed so that it can be hosted by independent service providers (ISPs) as well as by Butterfly.net, the decision also had to be balanced by what these solution providers could deploy and support.

Maximizing the way online games are developed, deployed, and distributed keeps the accessibility and cost of playing these games down. Based on all of this, Butterfly.net thought that this software and hardware combination would be a natural, progressive choice for them.

Along with the hardware and operating decisions, Butterfly.net also examined approaches for effectively scaling and distributing system resources on-demand using the grid for game development. For instance, access to computing resources on demand allows game developers to avoid huge up-front infrastructure investment costs. The same is true for initial testing and game deployment. Because the grid can scale with a released game's growing user base, scaling the game on a grid allows both developers and producers to keep their cost structure relatively predictable.

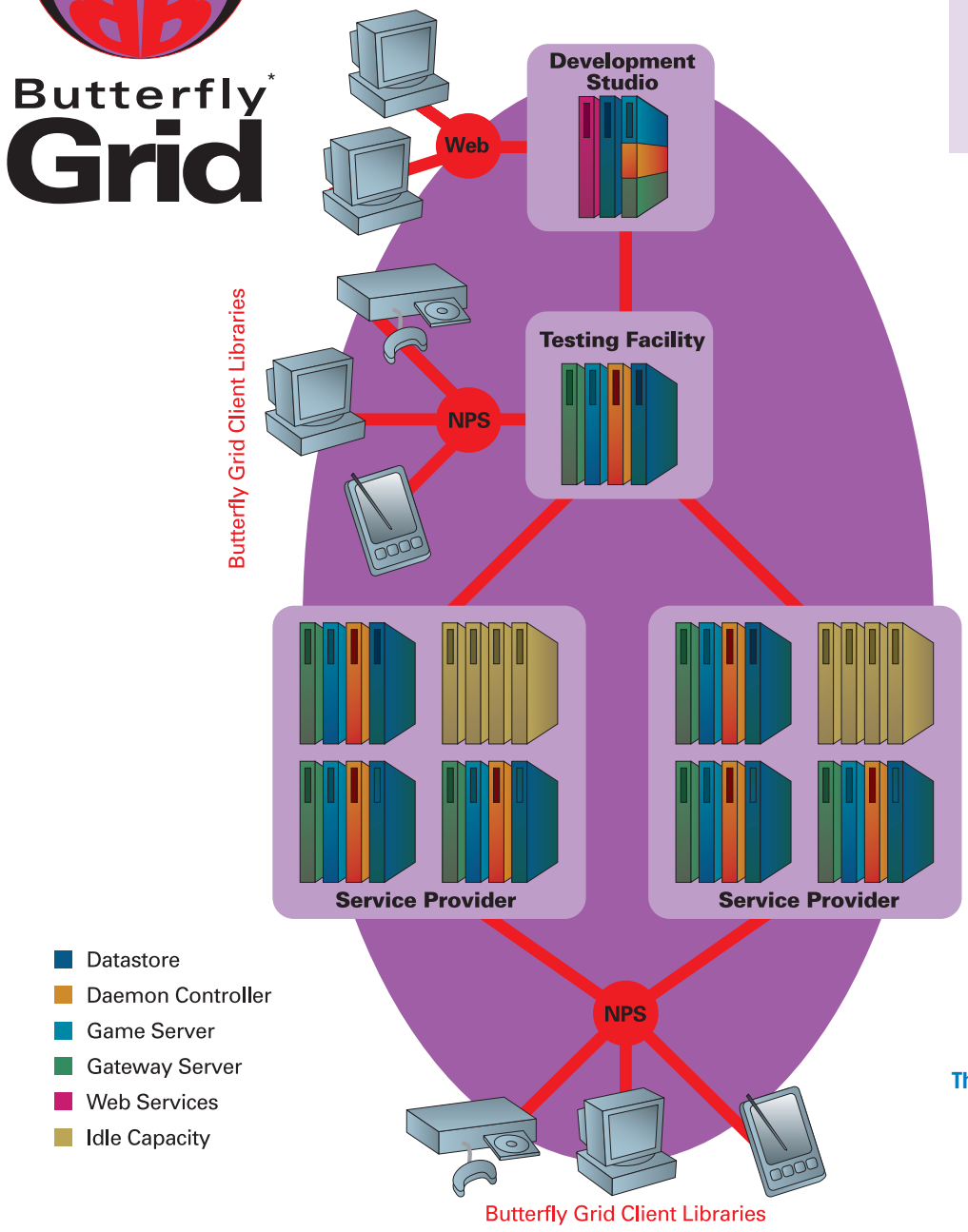
"Maximizing the way online games are developed, deployed, and distributed keeps the accessibility and cost of playing these games down."

The grid's usage-based cost structure increases return on investment (ROI) for game developers, and allows greater pricing flexibility for mass-market adoption, faster time-to-market, and quicker volume delivery.

The grid that Butterfly.net developed provides a way to use both technology and operational investments on the same platform. When fully deployed, the Butterfly grid architecture can support over one million simultaneous players without compromising performance.

Butterfly* Grid™

> "The grid that Butterfly.net developed provides a way to use both technology and operational investments on the same platform."



The Buterfly Grid Architecture

Under the Hood: Closer to the Metal

As mentioned earlier, game developers call the art of optimizing code for a particular game platform getting "close to the metal." Optimizing a game and getting close to the metal when the game is on a self-contained computing device, such as a personal computer, console, or arcade machine, is a different challenge than writing tight code for an MMG, which must support hundreds, thousands, and even millions of gamers interacting with one another and with non-player characters (NPCs) over the Internet. Some of the MMG development considerations are:

- The various client devices, the servers, and the network
- The location where game logic is executed: on the client's machine or on the server
- The timing of updates for the clients and the servers
- The timing for writing to the database
- The timing and location of information storage

All of these must be balanced to provide optimal game play.

These challenges in designing and building MMGs resulted in Butterfly.net building an innovative MMG platform that allows developers to get close to the metal—even when that metal is scattered throughout the world and across the Internet.

> "Optimizing a game and getting close to the metal is a different challenge than writing tight code for an MMG."

As gamers and software engineers, they believe that the only way for MMGs to enter the mass market as a popular medium along with film, television, books, magazines, and radio, is for game designers to have a platform to write to, and for the platform to meet the following key requirements:

- Utterly reliable
- Infinitely scalable
- Absolutely secure
- Industry standard
- High performance
- Maximally efficient
- Exceptionally entertaining

Utterly Reliable

To create a truly reliable system, the servers themselves must be hot swappable. If a server is taken offline, another server must be ready and able to take its place. If a hosting center or a section of the Internet is not available, connections to the game must be re-routed to a new set of resources. This must be unnoticeable to the gamer. Reliability dictates that the game architecture be fully distributed, and that the data that comprises a persistent state be instantiated from many different databases into many different game servers and written back to redundant data-stores as game play progresses.

Reliability also dictates that the processes by which game play is distributed among servers be automated. System administrators can be reacting to changes in game play and looking for resources to acquire. A truly autonomic computing infrastructure is required. These requirements led to the selection of hot-swappable blade technology based on Intel Xeon processor architecture.

Infinitely Scalable

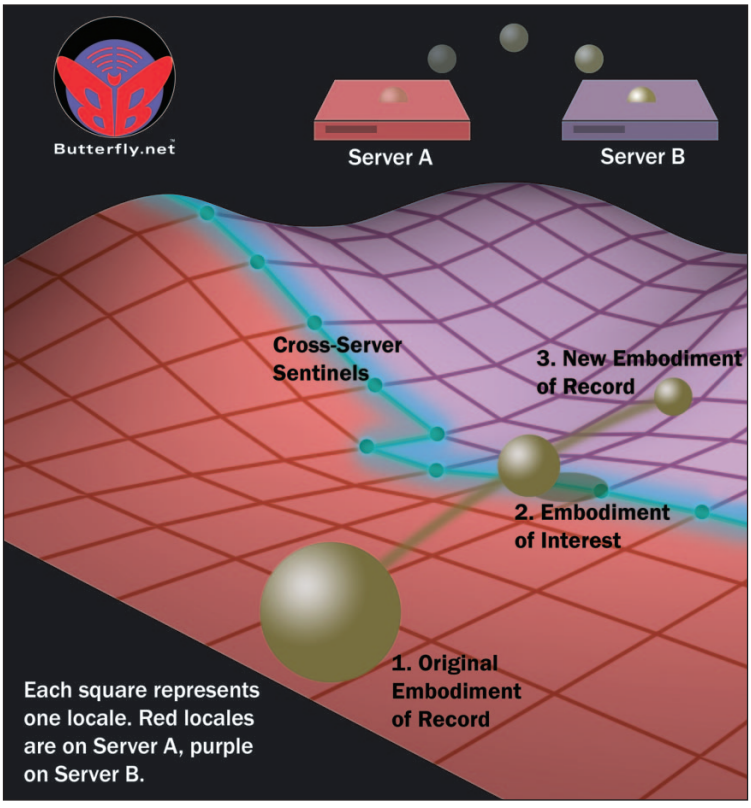
Today's games are not built to easily scale. Most game publishers simply replicate worlds on multiple systems to create scale. This inevitably leads to problems because MMOG players are then bound to specific game servers, with each server able to support only a finite number of players at any one time. The games themselves do not truly scale, the systems merely do.

Currently game operators have the choice of building large-scale infrastructures to support anticipated player numbers. Building infrastructure represents a large initial investment that is at risk if the game does not attract the expected audience. Worse, if the game attracts more players than anticipated, inadequate systems may not be able to handle peak loads or over-subscription periods, which could result in poor game performance or worse, a halt altogether.

A fully distributed game infrastructure, such as the Butterfly grid, scales by allowing the game to acquire more computing resources as needed, and by flexibly dividing up game play, to best take advantage of the systems available. On the Butterfly grid, the game world is not replicated on the servers, but portions of the world are placed on different servers so that the game itself scales.

The server infrastructure itself can be heterogeneous. Many 1U rack-mounted Intel Xeon processor servers can inexpensively be added to the grid, as well as powerful computer clusters or high-end servers based on Intel Xeon processors using blade technology. The scaling model that makes the most sense for the operations team can be accommodated. While a developer can still segment players and games into shards, it should not be mandatory because of server constraints. A single shard can run across many servers so that resources can be allocated to the area that requires the most processing power.

> "A fully distributed game infrastructure scales by allowing the game to acquire more computing resources as needed."



A Fully Distributed Game Infrastructure

> "The game industry is a long way from standardization, but the Butterfly* grid is a step in the right direction."

Absolutely Secure

Security involves three things:

- Authentication
- Authorization
- Accounting

The Butterfly grid uses several security protocols, including a 128-bit session message digest that can only be generated by an authentic client (one with both the session key and the account password). Rules enforcement is handled on the server, ensuring that players do not cheat by hacking into the system.

Industry Standard

The game industry is a long way from standardization, but the Butterfly grid is a step in the right direction. Using a set of supportable industry standards and architecting a distributed computing system specifically for games that takes advantage of standard components, games are designed and optimized for that system. The distributed infrastructure becomes the platform that the game developer writes to, rather than writing to the specifications of the edge device. They are working with the Global Grid Forum and use Open Grid Services Architecture (OGSA) components to ensure that any game that conforms to open standards and publicly available specifications operates over the Butterfly grid.

High Performance

The Butterfly grid is optimized for online gaming in several ways. The dead reckoning systems ensure that communications among clients and servers only occur when a model is not synchronized with others. The game developer controls the level of tolerance that triggers communication and resynchronization. In a first-person shooting, racing, or other action game, the objects can update each other frequently, and the game developer limits the number of objects within an area of interest. In an epic adventure, role-playing, or strategy game where thousands of players may be in one area, the dead reckoning models can be set to support an appropriate level of tolerance.

Dedicated servers within the grid run artificial intelligence as well. Games with lots of intelligent creatures can dedicate resources to running those creatures without bogging down the game servers. In addition, the network protocol stack itself is based on UDP with a thin reliability layer for optimal performance.

UDP sends each datagram in the Butterfly grid as an individual unit with no connection to other units that it sends. In contrast, TCP would send a stream of data, which is interconnected to the preceding and following packet. A packet of TCP data is not sent until receipt of the preceding packet has been acknowledged by the receiver. After a delay, it resends the missing packet thus ensuring the entire message gets through.

While this is a highly efficient model in a normal data-driven Internet protocol-based network, this can cause problems in a distributed game environment. TCP ensures that packet loss does not occur, but at the price of a heavy communications overhead, because any transmission loss necessitates the retransmission of the data. This inevitably leads to real-time delays in the transmission, which in gaming scenarios appear to the player as jitter.

UDP largely circumvents the connection setup process, flow control, and retransmission problem. UDP-based grids allow the sender to specify both the source and destination port numbers for their message. Coupled with a calculation of the checksum on both the data and header, this allows both the sending and receiving applications to ensure the correct delivery of a message without the overhead TCP imposes. In the Butterfly grid architecture, the player is a known, proxied entity, and so managing source and destination information can be efficiently handled using UDP. Transmissions between the player and grid components are more instantaneous, and thus preferable.

Maximally Efficient

The Butterfly grid is exceptionally efficient, allowing the game to allocate resources to the areas that need them the most. All server code is highly optimized C and C++ for Linux and client code hand-tooled to the various edge devices (such as PS2, Microsoft* Windows*-based computers, and Microsoft PocketPC-based devices).

In an MMG, it's not enough to optimize the network protocol stack, the client code, and the server code; all three systems must be synchronized and work together for the best overall effect. That effect also depends on the game itself. With the Butterfly grid, the game developer controls the clients, the server, the gateway, the artificial intelligence engines, and the database management system, and can make the most intelligent trade-offs to optimize the gaming experience, while using the fewest resources (client processor, bandwidth, and server capacity) overall.

Exceptionally Entertaining

Architecture requirements led Butterfly.net first to a fully distributed architecture, then to grid computing, and finally, to the Butterfly grid. With this platform, developers could finally optimize code for the Internet itself, creating games that incite, confound, baffle, delight, and amuse gamers.

Fully-Meshed Metal

The Butterfly grid architecture has the potential to fundamentally change the online gaming community, and change the way online games were developed and deployed. Today, developers work on multiple platforms and then deploy on others. Leveraging network costs over multiple titles rarely happens today. Developers focus on one game, and have yet to provide an acquiring publisher with a way to share capacity, and thus cost, across numerous titles.

To overcome these challenges, a solution needs to:

- Provide code optimization levels that developers are used to working with.
- Be game-agnostic.
- Offer open-source flexibility for certain aspects of the system, such as the network protocol stack, cross-server message sources and sinks, state management systems for the server and network, gateways to transfer players from one server to another seamlessly and transparently, and the artificial intelligence subsystems. The open source Linux operating system was chosen to satisfy this requirement.

"The Butterfly* grid architecture has the potential to fundamentally change the online gaming community, and change the way online games were developed and deployed."

To create a software system architecture that responds to these challenges, Butterfly.net had to decide whether to create the architecture entirely themselves, or integrate portions of it from existing services and focus on adding value for the MMG market. They selected the latter approach and added value by creating sophisticated software structures to support MMG within a grid.

Linux was the chosen operating system. Butterfly.net also developed a middleware application server core that could support multiple functions within the software architecture. Besides connecting to the system's back-end database, which held all of the game's relevant data, the middleware core also had to maintain user persistence within the game's world.

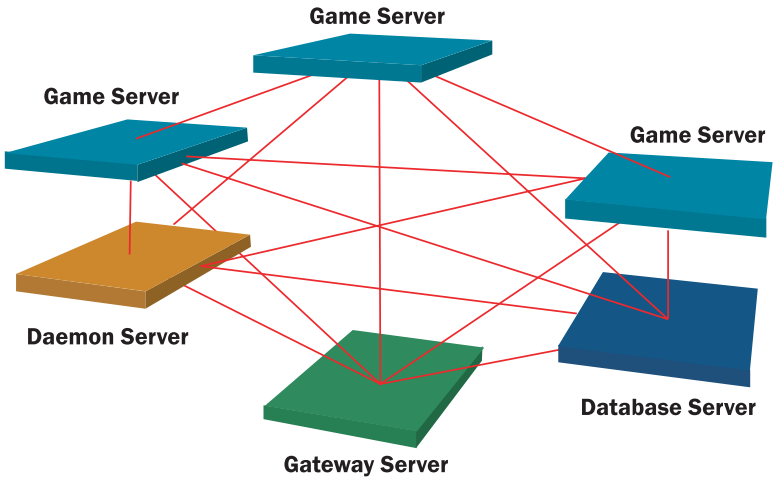
> "Butterfly.net developed a middleware application server core that could support multiple functions within the software architecture."

Butterfly's middleware core is based on a C or C++ architecture, highly optimized for the Intel architecture platform—and specifically the Intel Xeon processor—for all in-game functions. Business functions and grid interfaces are wrapped in Java*, with a Java-based messaging system for process management. They also incorporated a Java 2 Enterprise Edition (J2EE*) application server, which provided them with a sophisticated engine to handle real-time transaction volumes as users traversed worlds and shards.

Each mesh is connected over grid-compliant protocols to add services and computing capacity on demand. The Globus Toolkit*, a reference implementation of OGSA, provides the following services:

- Security
- Authorization
- Authentication
- File transfer
- Access to secondary storage
- Web-services bindings
- Meta-computing directory service
- Other standard utility services

The Butterfly.net Game Configuration Specification is an extensible markup language-remote procedure called (XML-RPC) Web Services Description Language (WSDL) binding, which is used by the toolkit to configure resources to support specific games.



Fully-Meshed Architecture of the Butterfly Grid

OGSA 3.0 Toolkit is a community-based, open-architecture, open-source set of services and software libraries that support computing grids and grid applications. Butterfly's use of specific toolkit components that are most relevant in their grid design include the grid resource allocation and management (GRAM) protocol and its gatekeeper service, which provides for secure, reliable service creation and management. The meta directory service (MDS-2) provides for information discovery through soft state registration, data modeling, and a local registry (called the GRAM reporter). The grid security infrastructure (GSI) supports single sign-on, delegation, and credential mapping.

The GRAM protocols mechanisms are used for authentication, authorization, and credential delegation to remote computations within the grid's distributed application base. A two-phase commit protocol is used for reliable invocation, and service creation is handled by a small, trusted gatekeeper process. The GRAM reporter monitors and publishes information about the identity and state of local computations using a registry within the grid.

MDS-2 provides a uniform framework for discovering and accessing system configuration and status information such as:

- Compute server configuration
- Network status
- Locations of replicated datasets within the grid hierarchy

MDS-2 uses a soft-state protocol for lifetime management of published information.

The public key-based grid security infrastructure (GSI) protocol provides MMOG players with single sign-on authentication, communication protection, and some initial support for restricted delegation. Single sign-on authentication allows a user to establish who they are once. The grid architecture then creates a proxy credential that it uses to establish the user's identity with any remote service on the user's behalf.

This is especially useful as gamers traverse shards and worlds within the grid itself, which can be physically distributed across a mesh of systems in differing geographies. Credential delegation allows for the creation and communication to a remote service of delegated proxy credentials that the remote service can use to act on the user's behalf—perhaps with various restrictions; this capability is important for nested operations.

The Globus Toolkit is the standard for grid-based sharing of online resources across organizations. It allows Butterfly.net to monitor servers and distributes the processing needs of more popular games and populated areas to idle computing resources within the data center. The key network protocol stack (NPS) is a thin reliability layer on UDP that connects edge devices to the gateways, which transparently relay information to the correct server.

Butterfly also uses multicast communications to all servers subscribed in the grid to create a fully meshed, multicast network during real-time game play. Additionally, each mesh is connected over grid-compliant protocols to add services and computing capacity, as needed, using OGSA components.

"The Globus Toolkit allows Butterfly.net to monitor servers and distributes the processing needs of more popular games and populated areas to idle computing resources within the data center."

Playing Games on the Grid: Architecture Components and Interaction

The Butterfly grid connects personal computers, notebooks, Pocket PCs, Palm*-compatible handhelds, and next-generation 128-bit consoles in one seamless world. An innovative packet transport protocol provides fast, balanced game play over broadband, dial-up, and mobile Internet connections for unique multi-channel interactions.

From the perspective of a game player, the software runs, the user log-on appears, and then the universe of the game opens. No shards, artificial constraints, or boundaries are visible. It appears seamless to the player, and whenever something seems seamless, you can bet that complicated actions are taking place behind the scene. That is the Butterfly grid. The magic takes place in the interaction between the protocol stack on the client, the gateway to which the user is connected, and the fully meshed server grid.

The Butterfly grid architecture is an n-tiered system incorporating multiple tiers, front to back.

The client libraries and their application programming interface (API) embodiment, called the object management system (OMS), comprise the first tier of the architecture. This tier is the window into the shared environment.

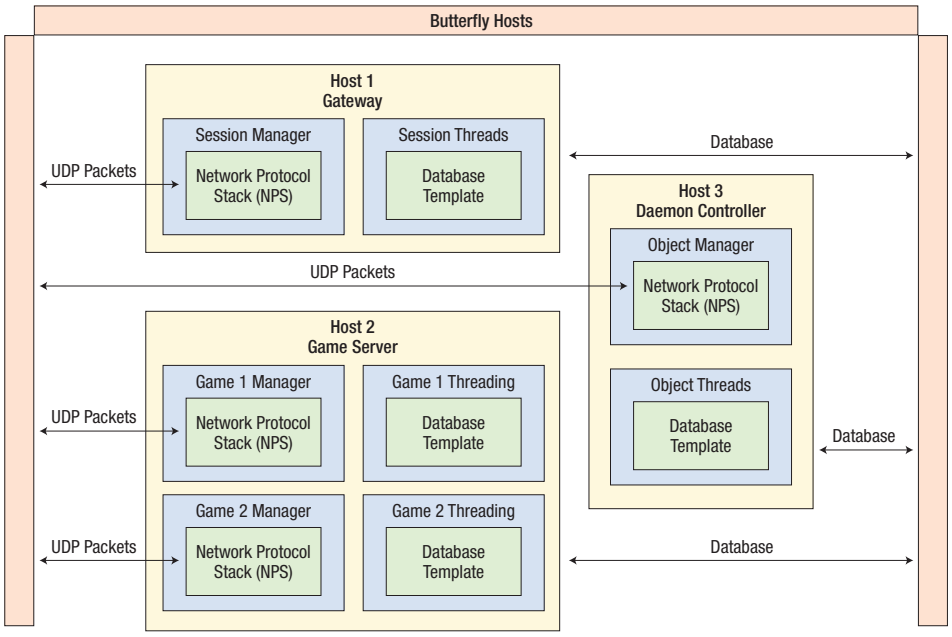
The client software renders data that has been translated to inherent protocols and supplied by the middle tier. Using the Butterfly network protocol stack, client software accesses system information via OMS, which is the interface to various server layers within the grid.

> "The Butterfly* grid connects personal computers, notebooks, Pocket PCs, Palm*-compatible handhelds, and next-generation 128-bit consoles in one seamless world."

The middle, or gateway tier, translates the data objects and communications protocols into forms that are understood by the user's platform (PS2, Microsoft Windows-based computers, and Microsoft PocketPC-based devices). It also handles and translates the interactions, changes, and actions of these game objects to communications protocols that are understood by the end-user's client platform. On a sufficiently complex or powerful platform, this layer can be thin. On more modest platforms, this layer may be complex and could involve different translations, where certain data elements are parsed out and not transmitted to the end client.

The back-end tier is the data store and server. This layer provides the embodiment of the game as it is defined by the game rules and logic, the objects and attributes that comprise the game environment, and objects that represent the players themselves as they are involved in game play.

Going in the other direction, game software provides natural interfaces for performing actions. These game actions are captured by the OMS, which in turn sends the information to the middle tier. The middle tier translates the information and communicates with the back-end tier, and re-distributes information to other client platforms, as appropriate to the game design.



Software Architecture of Butterfly* Grid Hosts

The gateway servers translate the data objects and communications protocols to forms that are understood by the user's platform and routes player connections to game servers.

The daemon controllers are dedicated artificial intelligence servers that drive the activities of non-player characters (NPCs). NPCs are game elements not directly controlled by player actions. They are essentially privileged clients that act like players and interact directly with the grid's gateway servers.

The game servers are responsible for running games within the grid. They manage the game as it is defined by the game rules and logic, the objects and attributes that comprise the game environment, and objects that represent the players themselves as they are involved in game play. The intelligence that determines when players are shifted to new servers resides on both the game and gateway servers. When a game server becomes overused or fails, it sends a controlling message to the gateway servers. The gateway servers are ultimately responsible for redirecting players to a new game server.

The database server stores the persistent information required to define the worlds and objects, and maintain game play over time.

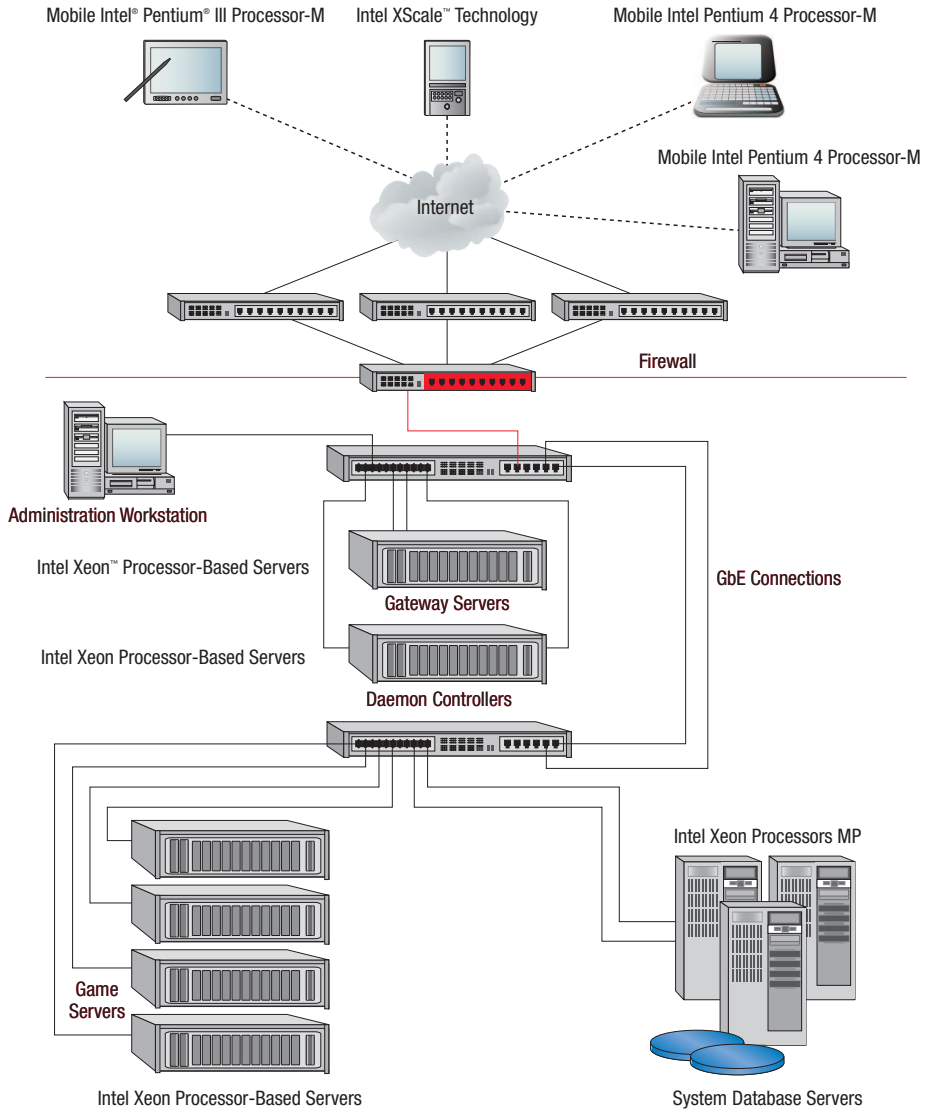
The network protocol stack is a thin reliability layer on UDP that connects edge devices to the gateways, which transparently relay on to the correct server. NPS is also used for real-time game play across server boundaries.

A multicast communication link connects game servers that are participating in the same game. These servers are fully meshed during real-time game play.

The utility computing layer is the lowest level of the infrastructure, monitoring the performance of the hardware and redirecting resources as required.

"The intelligence that determines when players are shifted to new servers resides on both the game and gateway servers."

Butterfly* Grid
Hardware Representation



> "When client software connects to the system, it can connect to any gateway that is in service."

When client software connects to the system, it can connect to any gateway that is in service. After authentication and authorization, the gateway translator acts as a proxy for the client to the back-end server. Multiple servers, each responsible for managing a segment of the environment can be used.

If in the course of using the system, the participant's state changes in such a way that they need to be served from a different server, a move request is transmitted to the gateway (a request that is generated by the server), at which point the gateway begins its proxy communications with the new server.

What's important is that as player state changes, and the gateway proxies the player to a different server, that server could physically be located in an entirely different geography or locale. Because the grid is a mesh, and that mesh is logically connected to other systems within the grid architecture, players can access grid components that are not physically located in the same place. Of course, this process is transparent to the end client device or user. A player does not even know it is happening.

The grid allocates resources based on usage, liberating game aficionados from server-bound game play. Once adopted on a global scale, the Butterfly grid will replace the far less efficient current online infrastructure and provide an open, scalable, high-performance environment that delivers responsive, reliable gaming action.

Summarizing Butterfly's Grid Architecture

The Butterfly grid is a powerful, innovative commercial computing development and distribution platform for the video game industry. An end-to-end solution for online video game developers, publishers, and service providers, the Butterfly grid can run multiple games and serve millions of gamers simultaneously—drawing resources from geographically dispersed servers and providing unprecedented gaming responsiveness.

The Butterfly grid delivers strong ROI benefits through cost savings fueled by more efficient use of computing resources and an increase in strategic flexibility. By relying on a grid-based MMP platform, game developers can avoid huge up-front infrastructure costs and procure only the computing resources that they need. This provides developers with significantly more latitude to pursue other flexible pricing and distribution models.

> "The Butterfly* grid delivers strong ROI benefits through cost savings fueled by more efficient use of computing resources and an increase in strategic flexibility."

Game developers find the grid offers an innovative way to build action, strategy, role-playing, simulation, and adventure games on a universal platform. Because the grid is engine-agnostic, it allows for more integration for both off-the-shelf and custom game engines.

Butterfly.net offers a unique license program that allows real-time prototyping on a live server grid with full bandwidth, simulation, and load testing. It allows developers to build games in a cost-effective, high-performance environment and provides the basis on the client side for processing the rules of play, individual players' avatars, and so on, through the thousands of end-user devices tied into the grid at any given moment.

Thanks to Intel architecture on the end-user side, the developer knows that when the code is written, tested, and deployed, the player experience will match what the developer had in mind. The Butterfly grid is rapidly becoming the global infrastructure for online games[3]. Service providers are adopting it to manage their resources more efficiently and create new revenue streams. Publishers are using it to leverage their technology investment while increasing profits, and developers are adopting the grid to build games quickly and economically without artificial server constraints. But the real driving force behind the Butterfly grid's growing popularity is the demand by gamers for a more realistic, fully immersive online experience.

[3] *Intel® Business Center Case Study*, "Butterfly.net Uses Intel® Architecture to Build a Global Gaming Grid", Pg.1, Copyright ©2003, Intel Corporation

www.intel.com/go/digitalmedia

251853—001