



*Distribute Your Applications and  
Mobilise Your Data*  
*White Paper*

**PROGRESS**  
SOFTWARE

*Real Time Division*

## *Table of Contents*

Table of Contents.....	2
Improving Access to Applications .....	3
Common Objectives & Requirement.....	3
Evaluating Options for Improving Application Access .....	4
N-Tier Client/Server.....	4
Advantages provided with this approach include: .....	5
Challenges presented by this approach include:.....	5
Thin Client “Application Access Portals” .....	6
Advantages provided with this approach include: .....	6
Challenges presented by this approach include:.....	7
Web Client .....	7
Advantages provided with this approach include: .....	8
Challenges presented by this approach include:.....	8
Distributed Applications & Data .....	9
No application redesign.....	9
Minimal bandwidth utilization .....	9
Low maintenance.....	9
Advantages provided with this approach include: .....	10
Challenges presented by this approach include:.....	11
Conclusion .....	11
About Progress Real Time Division .....	12



## *Improving Access to Applications*

The almighty “Corporate Headquarters” has steadily lost its gravitational pull over the past two decades. The advent of modern computer and communications technology has made physical proximity to central office infrastructure less and less important.

As a result, the workforce of today is more mobile and distributed than ever before. And, like the universe forming after the “Big Bang,” enterprises will continue to spread out to ever more distributed proximities in the future. Remote offices, regional and branch offices, international locations, home offices and mobile workers are already the norm today, and will continue to proliferate in the future.

The centralized architectures of most business applications today are completely out of sync with the distributed enterprise. These architectures provide maximum support for the main office, some support for remote offices, and hardly any support at all for mobile workers. The fundamental problem is that the Internet is being used in a way for which it was never designed: as the backbone for business applications.

There is no doubt that the Internet has been a breakthrough in terms of providing distributed workers with near-universal access to business applications in centralized data centers. However, it is still a highly unreliable and underperforming network, when compared to the typical corporate LAN, and is wholly sub-optimal as a backbone for business applications. All you have to do is talk to someone who works in a remote office, and you’ll hear them express frustration over application performance, availability and overall usability.

So, what can be done to align application architectures with the trend toward ever-more distributed enterprises? How can companies improve application access, performance and reliability for remote offices and mobile workers? There are several ways in which to provide improved access to applications and data. The following sections will detail the requirements companies need to consider when selecting the best option for doing so.

## *Common Objectives & Requirement*

In planning and evaluating options for improving access to applications and data for distributed users, there are several common and essential high-level requirements to consider.

Perhaps foremost, the solution must insulate the remote office from network outages and reduce the reliance on constant network connectivity. Applications and data must be buffered from network delays, uncertainty and outages, and be made more reliably available for distributed users. Given the Internet’s inability to provide reliable access, its role in the application architecture must be changed from “mission-critical backbone” to “occasionally needed service.” In other words, constant connectivity must be eliminated as a requirement for application usage, and replaced by “sometimes connected.”

Thus, remote offices and mobile workers must be able to operate while disconnected from the network. Enabling disconnected use of fully functional applications and data is an *essential* requirement for any distributed approach. And this does not mean providing users with read-only versions of their data. It means fully functional, read-write access to data – as if they were still connected to the network.

The solution should provide the access outlined above without degrading application performance. Application latency is typically a major source of frustration for users of

remote applications. Network outages and connectivity issues aside, the fact that an application must communicate via a network during its operation introduces very noticeable delays in processing and application usability – even over high speed networks. The effect is compounded as the distance between user and data center increases.

## *Evaluating Options for Improving Application Access*

As discussed earlier, the growing distributed nature of businesses requires a fundamental change in the way business applications are architected. From mainframe to client server to Web interface, application centralization and application distribution approaches have waxed and waned. Indeed, today's typical centralized Web deployments look remarkably similar to old monolithic mainframe deployments. However, there is a growing realization today that this mode of application deployment is untenable in a globally distributed world. Applications simply need to be deployed closer to the people who depend on them.

With the evidence of increasingly distributed business and the aforementioned guidelines in mind, let's review several approaches for deploying applications for improved access.

- 1) N-Tier Client Server
- 2) Thin Client "Application Access Portals"
- 3) Web Client
- 4) Distributed Applications & Data

### *N-Tier Client/Server*

This approach is closest to the traditional, in-house, centralized application environment. This scenario involves a central database server at the home office and deployment of robust client applications at each remote office or user location. The business logic may be contained within the client or may exist on a separate application server, located at the central office.

Companies that adopt this approach typically need to fortify their database server to support the additional remote users. It's not uncommon to purchase a scalable multi-processor server with at least half the number of possible processors and physical memory. This allows future expansion with minimized disruption. For example, a Sun Fire 6800 server with 16 out of a possible 24 processors and 16 GB out of a possible 192 GB of main memory would cost about \$850K and support thousands of users.

If a separate business logic layer is utilized, then the same capacity considerations would be given to the application server.

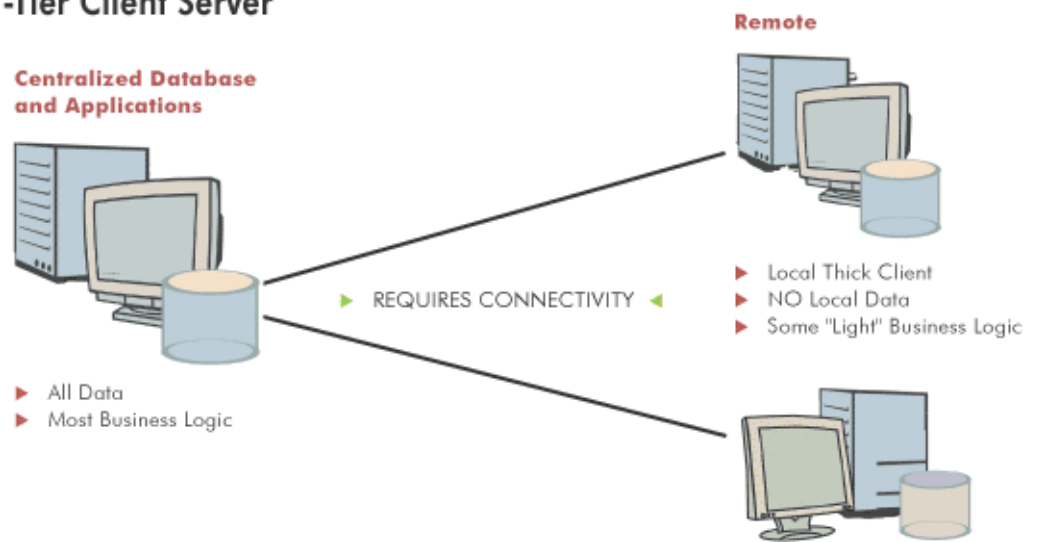
Typically, this architecture grows out of local application deployment, so the client application may need some modification to account for remote deployment.

Clearly, the lifeblood of this approach is network connectivity. Performance is directly tied to available bandwidth. A compromise must be struck between buying the highest available bandwidth and resigning oneself to potentially unacceptable performance during peak hours of usage. Since the application is making queries to a remote database server, large result sets can dramatically increase bandwidth utilization.

Besides performance, reliability is another concern. Without a connection from your remote users to the data center, the application cannot function. If your enterprise cannot tolerate anything beyond momentary periods where applications are unavailable, then consideration should be given to constructing a redundant, fault-tolerant network infrastructure. At a minimum, this would mean leasing two T1 lines at a cost of \$1,000 per month from different providers (for risk isolation) and a fail-over bridge-router for 'hot' switch-over. To improve the reliability of the central site, a failover system would

be needed that could provide minimum application support for core set of users at a cost of \$450K to \$850K.

## N-Tier Client Server



*N-Tier Client Server Architecture*

### Advantages provided with this approach include:

- ⇒ One database with centralized management
- ⇒ One application server with centralized management
- ⇒ Adding new users is straightforward
  - Install client software on a device
  - Pointing to the existing central server

### Challenges presented by this approach include:

- ⇒ Network dependant architecture
  - Central network infrastructure failure is visible immediately to all users
  - Regional network infrastructure failure impacts all region's users
  - Bandwidth intensive and limited to capacity of central pipe
  - Once the bandwidth limit is reached all users have reduced quality of service
  - Subject to latency impacting user experience
- ⇒ Dependent upon central server(s) at central office
  - Single point of failure at server or data center level
  - Limited to failover model – cannot load balance database work
  - Costly server hardware with idle failover capacity
  - Limited maintenance window –especially for global enterprises
- ⇒ No support for disconnected operation

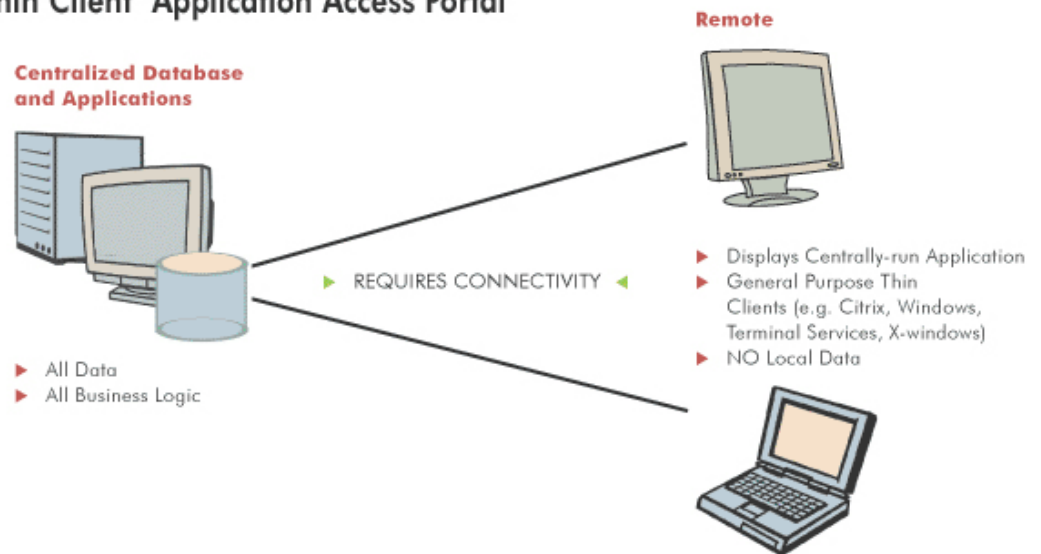
## Thin Client "Application Access Portals"

This solution, effectively, represents remote control operation where network dependant terminals access one or more central servers via Citrix's ICA or Microsoft's RDP protocols. Each user has their own virtual machines running on these central servers, on which the applications are loaded and executed.

Due to the virtual machine nature of this environment, an application must be qualified to work in this partitioned mode and may possibly require changes. For applications where the source is under the enterprise's control, this may not be an issue. But for commercial applications, this can present a significant obstacle. Bandwidth utilization is minimized since all queries are local to the data center server(s) and application user interface elements are tokenized between the server and the remote client user. Regardless of bandwidth requirements, network connectivity is still required in order to use the application. Therefore, the same network reliability issues associated with the n-tier client-server approach still apply. Network latency may be more of an issue with this approach, as keystroke and GUI data must be sent between the thin client and the server.

Because the servers must be able to support the load of a virtual machine for each user, the resource requirements are quite large. The multi-processor server choices for traditional Windows-based hardware are rather limited, compared with that of other server architectures. For example, the Dell PowerEdge 6600 server tops out at 4 processors and 32GB of SDRAM, costing about \$100K each. Typically, several servers would be needed to host the application-presentation layer, in addition to database and application servers that could lead to hardware costs well over \$1M to support a thousand users.

### Thin Client "Application Access Portal"



*Thin Client "Application Access Portal" Architecture*

### Advantages provided with this approach include:

- ⇒ Standard configuration of application on central servers
- ⇒ Centralized database and application server resources
- ⇒ Quick deployment with minimal software on clients

## Challenges presented by this approach include:

- ⇒ Applications must be compatible with multiple virtual machine partitioning
- ⇒ Windows-only solution for projecting user interface to remote PCs.
- ⇒ Latency issues for interactive operations
  - All keystrokes and GUI data transferred
- ⇒ Network-dependant architecture
  - Central network infrastructure failure is visible immediately to all users
  - Regional network infrastructure failure impacts all region's users
  - Bandwidth-intensive and limited to capacity of central pipe
  - Once bandwidth limit is reached all users have reduced quality of service
  - Subject to latency impacting user experience
- ⇒ Tremendous server loading – local PC is idle and resources wasted
- ⇒ Dependent upon central server(s) at central office
  - Multiple, expensive servers for GUI, application server, and database
  - Single point of failure at server or datacenter level
  - Limited to failover model – cannot load balance database work
  - Costly server hardware with idle failover capacity
  - Limited maintenance window – especially for global enterprises
- ⇒ No support for disconnected operation

## *Web Client*

The Web-based approach has grown out of the proliferation of Internet technologies during the 1990s and encompasses several different client implementations, the most common of which are:

- ⇒ Web browser based, using locally executed HTML, Java and JavaScript
- ⇒ Server-deployed but locally executed Java Applets, or browser plug-ins

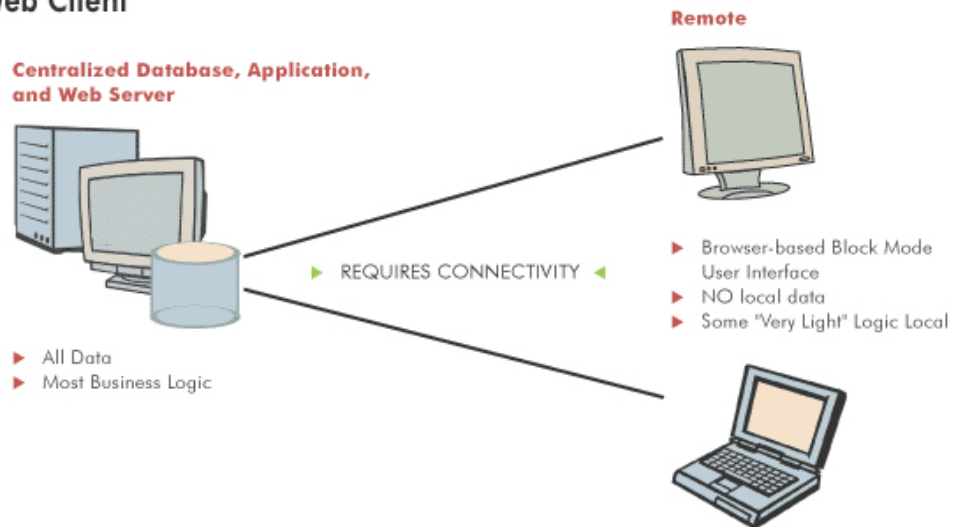
More recently this has expanded to include 'Thick Client'-based applications using Web services technologies (J2EE, .NET) and blurs the line between Web client and client/server approaches.

Regardless of the client implementation technology, all approaches rely on network connectivity, as well as the presence and performance of one or more Web and/or application servers.

Unlike the thin-client approach, interactive latency is not usually a problem, as all GUI functionality takes place on the remote user's system. However, general application performance is bound to suffer, since queries and returned data must be exchanged between the user and the data center server. Therefore, the same bandwidth considerations identified with n-tier approach will also apply to the Web client solution.

As with all the approaches mentioned thus far, network and server reliability is the determining factor for application availability. This situation is exacerbated with Microsoft's .Net technology, in that the dependency upon multiple services can be distributed among multiple remote servers. This multiplies the complexities of managing the "single point of failure" risks.

## Web Client



*Web Client Architecture*

### Advantages provided with this approach include:

- ⇒ Application is deployed to client at execution time
- ⇒ Can build generic application that runs on multiple clients (Java)
- ⇒ GUI performance is not impacted by network latency
- ⇒ Centralized management of database and application servers

### Challenges presented by this approach include:

- ⇒ Existing applications must be completely re-written using new technologies
- ⇒ Limited application functionality with browser based clients
  - Very limited local storage of working application data
- ⇒ Network dependant architecture
  - Central network infrastructure failure is visible immediately to all users
  - Regional network infrastructure failure impacts all region's users
  - Bandwidth intensive and limited to capacity of central pipe
  - Once bandwidth limit is reached all users have reduced quality of service
- ⇒ Dependent upon central server(s) at central office
  - Single point of failure at server or data center level
  - Limited to failover model – cannot load balance database work
  - Costly server hardware with idle failover capacity
  - Limited maintenance window –especially for global enterprises
- ⇒ No support for disconnected operation

## *Distributed Applications & Data*

Application and data decentralization has become a major trend in the industry. When applications are deployed in this way, remote offices do not shut down when network connectivity is slow or lost. Mobile workers can bring their applications into the field with them when they need them most. And the data center is no longer a single point of failure in the enterprise.

This approach involves deploying independent, replicated database instances along with a robust client application, either in a remote office or on a user's laptop. While not necessarily requiring "real-time" data mirroring, the database needs to be synchronized at regular intervals, the frequency being dependent on the application and business requirements. This solution has a distinct advantage in that it can tolerate frequent network outages and bandwidth restrictions and still allow remote users to continue working. Additionally, off-the-shelf commodity computers can be used, thereby reducing the need for expensive, multi-processor servers at the data center. Rather than providing all access to the data and business logic from a central location, that processing is distributed to smaller servers across the enterprise. Centralized systems may still be needed for reporting purposes or to service large sites.

Depending upon the technologies chosen to construct this solution, there is a potential for several disadvantages:

- ⇒ Extensive application redesign
- ⇒ High bandwidth utilization
- ⇒ High maintenance burden

However, constructing a distributed enterprise with the Progress® DataXtend™ RE Distributed Enterprise product suite eliminates these problems:

### **No application redesign**

Some queue or message-based middleware solutions require that special APIs be used in order for an application to communicate with the database.

DataXtend RE Distributed Enterprise allows you to communicate with the database as you normally would, using any programming language currently in use. The result is that no application changes are needed to access the database. And because DataXtend RE uses "update everywhere" technology, no application changes need to be made in order to update data. This is not necessarily the case with master-slave or synchronous update architectures. It may still be necessary to modify the application to support distributed algorithms replacing, for example, a portion of the application that hands out sequential order numbers with a method that can operate in a distributed environment. Generally these changes are straightforward and well understood by the application developers.

### **Minimal bandwidth utilization**

Most replication solutions send the full content of all transactions to each site including all intermediate changes to the same data. Generally the complete dataset must be maintained at all sites.

DataXtend RE uses a "net change" model, so only the data that has been updated since the last replication is sent through the network. Additionally, DataXtend RE technology allows you to partition data so that only the information pertinent to that site is sent through the network.

### **Low maintenance**

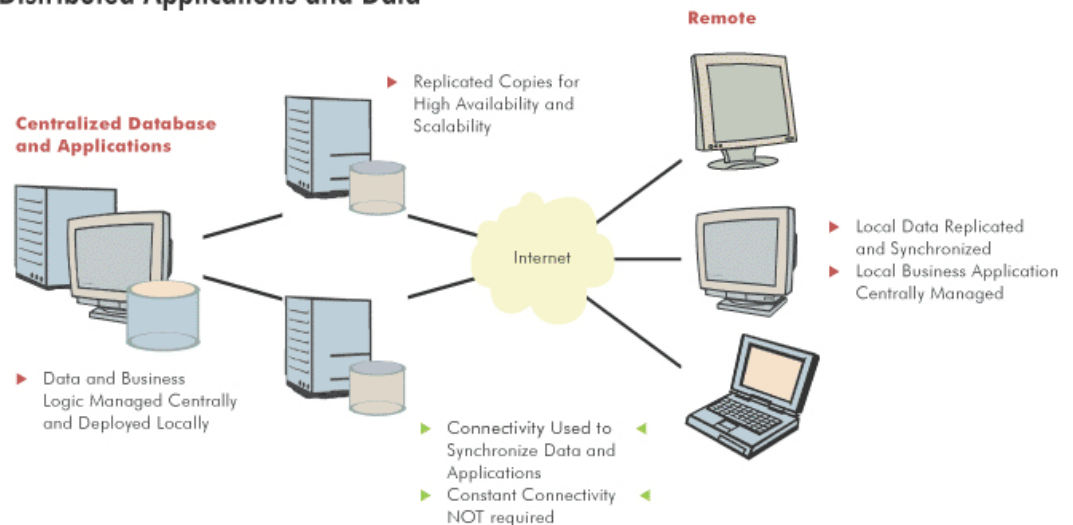
Log-based replication solutions need to be periodically synchronized. Additionally, when sites haven't replicated for extended periods of time, the log file may fill the server's disk and result in downtime and an unscheduled synchronization session.

DataXtend RE Distributed Enterprise takes an entirely different approach using “live” database access during replication sessions, so no synchronization is needed and there are no log files to manage. DataXtend RE-based applications can operate in a disconnected environment for extended periods with no additional maintenance requirements.

Other replication solutions require extensive administrative intervention to resolve data conflicts. This is because the typical unit of replication is an entire data record.

DataXtend RE technology allows administrators to partition records by update authority, allowing simultaneous updates when it complies with your business rules and avoiding false conflicts.

## Distributed Applications and Data



*DataXtend RE Distributed Applications and Data Architecture*

## Advantages provided with this approach include:

- ⇒ Isolated from network outages
- ⇒ Independent of central server “down-time”
- ⇒ Uses commodity, off-the-shelf hardware
- ⇒ Supports occasionally connected users
- ⇒ Supports occasionally disconnected users
- ⇒ Always optimal application performance
  - Even when disconnected
- ⇒ The same application can run everywhere
- ⇒ Data partitioning
  - Efficient bandwidth usage
  - ‘Need to know’ security
- ⇒ Platform and Database independent
- ⇒ Centrally managed and distributed data

## Challenges presented by this approach include:

- ⇒ Minor application changes may be required
- ⇒ Configuration of replication network
- ⇒ Remote server, desktop, notebook PCs need to support local execution

## *Conclusion*

There are many technologies to consider when deciding which approach to use for your distributed enterprise. Most seem credible and many do, in fact, address some of the centralized infrastructure problems you are trying to solve. On the other hand, many approaches still have some of the same old dependencies that currently prevent your organization from realizing its full potential.

Only one of the approaches is a break with the past, designed specifically to support the needs of increasingly distributed organizations: the distributed data and applications approach. This distributed model gives all workers – whether they are in the home office, in a remote sales office, or sitting in their car with their laptop – equal access to perfectly performing and fully functional enterprise applications.

If you are attempting to create more distributed application architectures, DataXtend RE recommends that you clearly outline your goals and define your criteria for success. Then, re-examine the distributed enterprise technologies that are available to you and choose the one that best fits your enterprise needs, and provides the highest likelihood for success.

For further information, please refer to the other documents in the DataXtend RE Technical Library, especially the “Checklist for Evaluators of Data Replication Solutions”.



## *About Progress Real Time Division*

The Progress Real Time Division provides event stream processing, data management, access and synchronization products to enable the real-time enterprise. Our products manage and analyze real-time event stream data for applications such as algorithmic trading and RFID; accelerate the performance of existing databases through sophisticated caching; manage and process complex data in the industry's leading object database; and support occasionally connected mobile users requiring real-time access to enterprise applications. The Progress Real Time Division is an operating unit of Progress Software Corporation (Nasdaq: PRGS), a global software industry leader. Headquartered in Bedford, Mass., they can be reached at [www.progress.com/realtime](http://www.progress.com/realtime) or +1-781-280-4000.

**PROGRESS**  
SOFTWARE

**Real Time Division**

[www.progress.com/realtime](http://www.progress.com/realtime)

### Worldwide and North American Headquarters

Progress Real Time Division, 14 Oak Park, Bedford, MA 01730 USA Tel: +1 781 280 4000

### UK Office and Northern Ireland

Progress Real Time Division, 210 Bath Road, Slough, Berkshire, SL1 3XE England Tel: +44 1753 216 300

### Central Europe

Progress Real Time Division, Konrad-Adenauer-Str. 13, 50996 Köln, Germany Tel: +49 6171 981 127

### France

Progress Real Time Division, 3 Place de Saverne, Les Renardières B, 92901 Paris la Défense Tel: +33 1 41 16 16 56

© 2005 Progress Software Corporation. All rights reserved. Progress and DataXtend RE are trademarks or registered trademarks of Progress Software Corporation, or any of its affiliates or subsidiaries, in the U.S. and other countries. Any other trademarks or service marks contained herein are the property of their respective owners. Specifications subject to change without notice. Visit [www.progress.com/realtime](http://www.progress.com/realtime) for more information.