The Zack Edge Application Server

Technical Specifications and Network Integration

Whitepaper, July 2001

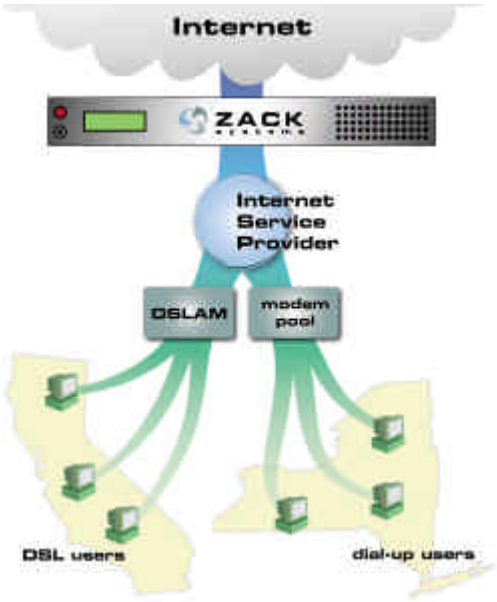
Zack Systems
191 West 25th Avenue
San Mateo, CA 94403

I. Introduction

Zack Systems has created a suite of Edge Application Servers ("Servers") designed for Internet Service Providers (ISPs) to enhance their relationship with users in order to drive new revenue opportunities. The primary function of the Zack Servers is to add, extract, monitor and modify information in the data stream. In effect, the Zack Servers provide a server-side mechanism to deliver value-added applications and tools that generate additional revenue from the user's experience online.

This document describes the technical architecture of the Zack Servers and how they are integrated into an ISP's existing network infrastructure. Figure 1 provides a simple schematic of where the Zack Server sits in the network relative to the ISP and its users.

Figure 1 — Zack's Position at the Network Edge



The Zack Servers sit at the edge of the network between the user's access point and the Internet to deliver enhanced services, ISP branding and advertising among other applications and tools.

II. Zack Architecture

A. Overview

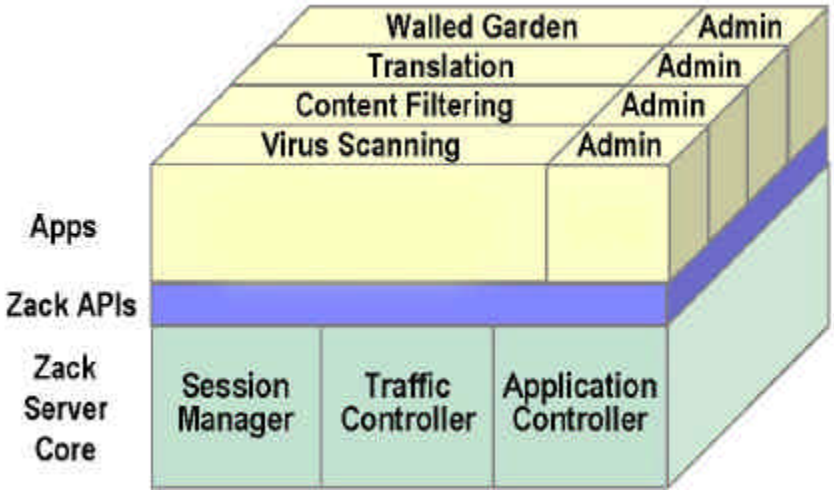
Zack Servers contain three core components: the Session Manager, Traffic Controller and Application Controller. These components work in concert to identify users, determine which applications a user has enabled and run those applications on data as it passes through the Zack Server (see Figure 2 below).

The Session Manager integrates with existing authentication infrastructure to reliably identify users. By interfacing directly with access servers, the Session Manager can determine which ISP account is associated with each traffic stream and route the data through the appropriate applications.

The Traffic Controller routes traffic through value-added applications based on a user's service level and application subscription status based on data provided by the Application Controller. The Traffic Controller interacts with existing network infrastructure to redirect traffic to the applications that the user has purchased. The Traffic Controller completes the transaction after the applications have processed traffic and performed the appropriate monitoring or modification tasks necessary.

The Application Controller manages application subscriptions and allows users to interact with applications through a variety of user interfaces. The Application Controller contains the primary repository of information on application subscriptions that are used to select which traffic should be routed through each application. The Application Controller also provides a mechanism for users to change their application subscriptions and application settings.

Figure 2 — Zack Server Architecture



The Server Core consists of the Session Manager, Traffic Controller and Application Controller. The APIs above this core directly influence the capabilities of the Zack Server to add, extract, monitor and modify information in the data stream.

B. Session Manager

Individual user identification is an important part of enhanced services, messaging and targeted advertising applications. The Zack Server addresses this need through a variety of components within the Session Manager that are specialized to manage user identification information within specific deployment situations.

1. NAS interface, RADIUS proxy and SNMP interface

In general, a Zack Server will be integrated with an ISP's Remote Access Servers (RAS) or Network Access Servers (NAS). The Zack Server can either query the RAS/NAS units via SNMP to obtain the required user identification information, or it can be configured to act as a RADIUS/AAA proxy, thus intermediating each authentication transaction between the RAS/NAS and the ISP's RADIUS/AAA server(s).

2. Heuristic user identification (optional)

For certain deployment situations where probabilistic user identities are sufficient, a Zack Server can employ a heuristic session manager, which uses both IP addresses and request signatures (browser type and request sequence) to track individual users. In this case no association is made between a particular user and the associated ISP account. This approach would be suitable in any case where precise user identification is not vital to the application performing properly (i.e., non-targeted advertising or one service tier for all users).

C. Traffic Controller

The Traffic Controller is the core of Zack's traffic-processing architecture. The Traffic Controller is a multi-protocol processing engine that can intermediate HTTP and other protocol requests with no perceived latency. The engine operates by multiple methods, including functioning as a transparent or explicit proxy and executing requests via Web-cache callout protocols such as Network Appliance iCAP or CacheFlow ACE.

1. Networking layer

- Requests supported: explicit-proxy, transparent-proxy and callout
- Configure as: either router or default gateway for layer 4 redirection

The Traffic Controller can handle explicit-proxy, transparent-proxy and callout-protocol requests simultaneously, affording maximum flexibility in deployment. In most cases a Zack Server will be deployed in conjunction with a layer 4 switch, but the Traffic Controller can also handle layer 4 redirection itself provided that 1) no load-balancing is required and 2) the Zack Server is configured as a transit router or the default gateway for the network in question.

Because the Traffic Controller uses a non-blocking, single-threaded architecture, it can handle a very large number of simultaneous requests with low incremental overhead per simultaneous connection. This helps the Zack Servers handle wide variations in traffic loads and accommodate peak loads and overloads smoothly. (Especially in ISP settings, peak hours result not only in high bandwidth loads, but also a much larger number of simultaneous requests.)

2. Pass-through mode

- Supports data pass-through if processing is unnecessary

On a per-transaction basis, the Traffic Controller can utilize a special low-level data pass-through mode if no processing is needed for a particular transaction. This allows the Zack Server to incur minimal overhead for transactions where no modifications or monitoring are needed. The same method applies to transactions that initially appear as candidates for modification or monitoring, but subsequently are found to require no special handling.

3. Concurrent protocol engines

- Payload protocols supported: HTTP, FTP, SMTP, POP and IMAP
- Callout protocols: iCAP, ACE
- Layered support for access to underlying protocols

Within the Traffic Controller multiple protocol engines run concurrently to handle intermingled requests of different protocols, including HTTP, FTP, SMTP, POP and IMAP. Support for additional protocols will be added in the future.

The Traffic Controller can also automatically recognize and appropriately handle HTTP encapsulated within Web-cache callout protocols such as Network Appliance's iCAP and CacheFlow's ACE.

Applications written to the APIs of the Traffic Controller can access features and events of surface protocols (e.g., HTTP) and underlying protocols (e.g., iCAP, TCP).

4. Application models supported

In traditional proxy architecture, a single inbound request corresponds to a single outbound request. Applications written to the Zack Servers are not limited to traditional "proxy" modes of operation, but can function according to many different models, including:

- Proxy: a single inbound request results in a single outbound request
- Server: a single inbound request is fulfilled locally

- Aggregator: a single inbound request is fulfilled with a response composed of the results of multiple outbound requests

- Translator: an inbound request is fulfilled with the response to an outbound request or requests made in a different protocol

This flexibility enables the construction of advanced applications that deliver tangible value to the ISPs and their users.

5. Event-passing architecture for plug-ins

- The "Taos" API forms the core of Zack's traffic-intermediation architecture

- Events can be generated by
  - Protocol engines
  - Shared-state values
  - Other plug-ins

- Plug-ins can run locally with a Traffic Controller core or remotely (in a different process space or on a different machine) via the "Cimarron" remote plug-in system

6. Synchronization and timing control APIs

Applications written to the Traffic Controller operate on events generated by one or more of the protocol engines in the Traffic Controller or by other applications. This programming model is similar to the event-driven programming models used in all modern graphical user interface systems. Events can represent changes in protocol state, changes in blocks of payload data or metadata or changes in the state of other applications.

A given application can choose to process traffic of specific protocols or all traffic passing through a Zack Server. Applications can register the particular types of transactions they wish to process according to protocol-specific criteria and can register the events they wish to process from each such transaction. Generally object-oriented addressing paradigms are used within the Zack APIs. Depending on an application's needs, it may choose to handle all transactions with a single-object instance, with one instance for each of several transactions or with one instance per transaction.

In response to events, applications can monitor or modify the associated underlying protocol data or metadata via additional APIs. In many cases these operations proceed as quickly as possible, but an application may also choose (via other API calls) to delay all or part of a transaction based on the fulfillment of other conditions, including conditions based upon other transactions. This ability to synchronize the fulfillment of one transaction with others is crucial for applications that aggregate data from multiple sources, as well as for applications that use external databases to authorize (e.g.,

permitted / included URI[1] patterns, payment tokens), redirect (e.g., choosing the nearest of several servers), or validate (e.g., virus scanning templates) transactions.

Because a single-threaded architecture, such as the one used in the Traffic Controller, can make some applications more difficult to write, the Zack Server can also run applications in fully threaded environments with traditional execution mechanisms.

A distributed execution mechanism ("Cimarron") facilitates applications running in different local and remote contexts without any changes to the application object code. This same mechanism also provides for load balancing of applications among several servers (which is important for high-overhead applications such as human-language translation) and handles fail-over between application servers.

7. Inter-application communication, shared-state abstractions

Inter-application communication mechanisms provide facilities for applications to cooperate with one another, via event and message passing, as well as via several forms of shared-state abstraction.

D. Application Controller

1. Billing Interface

The Application Controller provides a Billing Interface that exports information about application use to existing billing systems. The Billing Interface allows automated retrieval of XML-based export files that summarize and enumerate billing events (such as new user subscriptions and service level changes). Translation modules produced by Zack's Professional Services use these files to update the ISP's billing system so that accurate invoices can be generated.

The Billing Interface provides a generic, event-generation API that identifies service events significant for billing purposes. This includes service changes (e.g., user upgrades through the included sign-up pages) and subscription changes, but it also has a capability of including events for metered billing.

2. In-page user interfaces

The Application Controller contains a toolkit allowing users to interact with applications through in-page user interfaces. These user interfaces include a hovering watermark logo and a persistent navigation toolbar. Both user interfaces are data driven and extensible for new applications and functionality. As new applications are added to the system, the user interfaces adjust to reflect the new data.

[1] Universal Resource Identifier is the generic term for all types of names and addresses that refer to objects on the World Wide Web. A URL is one kind of URI.

The hovering watermark logo is a DHTML-based user interface that maintains its position in the browser window as the user scrolls around on the page. This user interface forms the basis of many in-page communication features such as ad insertion and persistent messaging. The ISP can control the behavior of this user interface by adjusting parameters such as the duration and frequency.

The persistent navigation toolbar is an integrated system that uses a traffic rewriting module and DHTML script to place a permanent navigation toolbar in a frame at the top of the user's browser. The ISP specifies the navigation links that are provided in this toolbar. The toolbar can also serve as the central access point for user settings.

3. Stand-alone user interfaces

Applications may require stand-alone user interfaces for application configuration and control. The Application Controller provides a framework to ease the development of new user interfaces and incorporate them into a standard look and feel branded for the ISP. This framework integrates applications into the enhanced services framework seamlessly.

The Application Controller builds an Application Console out of the administration interfaces provided by the applications. The ISP's application administrator accesses all application controls from one integrated user interface. This user interface allows the administrator to create a tiered set of service offerings as well as to set prices for service tiers and individual applications. The Application Console is dynamic and data-driven, adjusting its user interface according to the list of available applications.

Users at a given ISP also have access to a settings interface to adjust individual application settings, such as the parameters for content filtering for example. The Application Controller provides a framework that presents this user interface with the ISP's logo, which further reinforces the brand and links the ISP to the value-added applications being provided.

4. Tiered Services Manager

The Tiered Services Manager maintains the database of user service levels and application subscriptions. Service levels specify a list of applications that are included in the base cost, as well as pricing for additional applications. Using this functionality, ISPs can structure different service tiers that are priced differently based on the particular application bundle and advertising components.

The Tiered Services Manager can also be controlled directly by users themselves, if configured as such. This self-serve user interface provides the ability for users to change service levels and manage subscriptions to individual add-on applications. These user interfaces work in conjunction with automated marketing campaigns (see "Active Marketing Toolkit" below) to allow users to respond easily to messages introducing them to new features. All changes are propagated to the Billing Interface for future invoicing.

5. Active Marketing Toolkit

The Active Marketing Toolkit gives ISPs the ability to initiate multi-media advertising campaigns to introduce users to new services. The toolkit also contains several pre-packaged campaigns that ISPs can customize for segments of their user base, where the ISP determines which aspects of the campaigns should be triggered and for whom (i.e., free trials, banner advertising, pop-up alerts and email messages). Each of the messages includes an self-service sign-up mechanism for users to obtain the targeted service immediately.

III. Network Integration

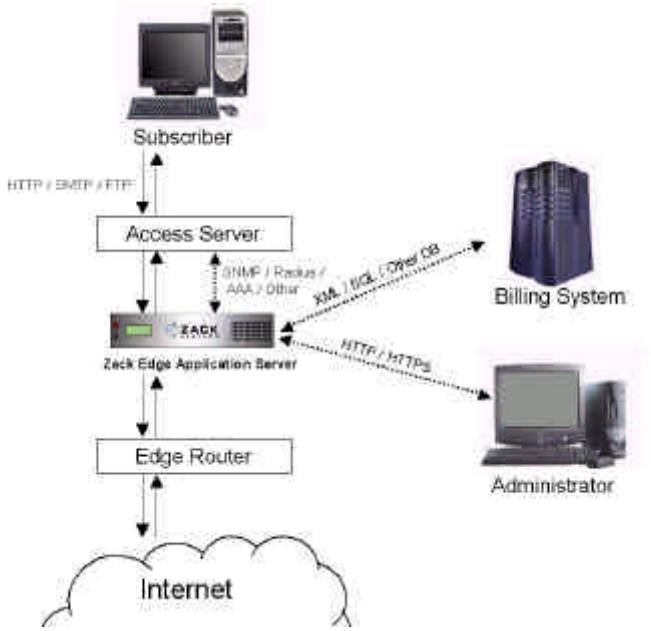
The Zack Server supports integration with existing network infrastructure using several different redirection mechanisms. These mechanisms integrate with standard components of the network infrastructure such as caches and layer 4 switches. The Zack Server processes traffic inside the ISP network at the same position as these components, between access server and edge router.

If cache integration is desirable, the Zack Server can be configured to receive traffic via the iCAP or ACE protocols (supported by Network Appliance and CacheFlow caches, respectively). These protocols redirect traffic to application servers as part of the caching operation. Through an alternate mechanism called "proxy chaining," the Zack Server can also be directed to route Internet-bound requests to the cache so that it can work with caches that do not usually support external application servers.

For high-performance deployments, layer 4 switches can be used to load-balance multiple Zack Servers. The layer 4 switch will ensure that traffic is distributed to all Zack Servers. The layer 4 switch can also handle failure conditions by redistributing the traffic to the remaining servers.

Low-profile deployments can use the Zack Server as a gateway, using the Traffic Controller to redirect selected traffic to Zack applications. In this deployment mode, no other redirection mechanism is required to give the Zack Server access to the ISP's traffic. This is ideal for smaller ISPs that do not wish to invest in additional switching capacity and those who do not have caches that support external application servers.

Figure 3 — Sample Network Integration



A sample of the Zack Server deployed in an ISP's network.

IV. Performance

A. Reliability and Fault Tolerance

- Stand-alone versus clusterable
- Layer 4 switch fail-over
- n+1 redundancy model
- Compartmentalized code failures
- Defensive programming
- Real-time-programming techniques ensure server responsiveness

The ability for a system to respond gracefully to an unexpected hardware or software failure is fault tolerance. There are many levels of fault tolerance – the lowest being the ability to continue operation in the event of a power failure. Fault-tolerant computer systems can mirror all operations – that is, every operation is performed on two or more duplicate systems, so that if one fails the other can take over. Another popular configuration is to have a “hot spare” operating at all times to take over the traffic capacity if one of the other systems fails.

The Zack Servers were designed to scale in an n+1 configuration where “n” is the system capacity necessary to service the user population, “N”. When a system is described as n+1, it indicates that there is an extra system component installed, brought online and receiving traffic as a “hot spare” (effectively, one unit of extra capacity).

Maintaining user-observed network performance is a primary design objective of the Zack Server. To this end, the Zack Server uses event-driven programming to preserve network performance and defensive programming practices to limit the scope of hardware and software failures.

Zack addresses the potential for hardware and software failure at several points within the Zack Server. The overall objective is that the impact of any given fault or failure should be as limited as possible and that traffic should continue to flow whenever possible.

First, the applications deployed on a Zack Server run within the context of a number of sanity-checking criteria, including running-time criteria, API-syntax criteria and memory-access criteria. Should any of these criteria fail, the associated application code will be disabled for the particular transaction involved. After repeated failures of the same application, that application may be disabled for all future transactions, while other applications continue to run.

Second, the core of the Traffic Controller continually monitors its own responsiveness through the use of deadman mechanisms. Should a fault be detected, the Traffic Controller will restart itself while continuing to accept new transactions and attempting to complete any in-progress transactions. If a serious fault persists, the Traffic Controller will either switch to a pass-through mode or cease operation entirely, depending on configuration. (The latter is more appropriate for any deployment involving a layer 4 switch.)

Third, Zack Servers are most commonly integrated with a layer 4 switch that already offers failure-isolation. If a Zack Server does not respond to HTTP requests, the layer 4 switch will stop sending requests to the Zack Server and route requests to other Zack Servers (if more than one is installed) or route them directly to the Web.

This soft-failure model strives to maintain the highest level of service for users, regardless of hardware or software problems.

B. Scalability

- Non-blocking, single-threaded core
- Automatic application load distributor
- Shared-state transaction mechanism
- Layer 4 load-balancing supports 30 to 50 Zack Servers in a pod
- Coping gracefully with extreme loads
    - Flow-control push-back
    - Load-shedding (optional)
    - Peak throughput capacity under load shedding exceeds 35 Mbps in current product and will exceed 200 Mbps in next release
    - Stand-down

Scalability refers to how well a hardware or software system can adapt to increased demands. For example, a scalable network system would be one that can start with just a few nodes in the cluster but can easily expand to thousands of nodes. Scalability is critical for an ISP because it means that when they invest in a system they will not outgrow it. Because the Zack Servers were designed to be linearly scalable, an ISP only needs to deploy additional nodes into the cluster to support more users.

To optimize scalability, the Traffic Controller employs a non-blocking, single-threaded core. This minimizes the incremental resource load of each concurrent transaction. Using this approach effectively removes most of the hard, concurrent transaction limits that threaded architectures can impose. The high-performance core is combined with mechanisms that perform high efficiency pass-through for objects that do not need to be modified. Most edge applications pass the bulk of transactions through unmodified, making this a valuable optimization.

In a multi-server deployment, any one of the Zack Servers can handle any transaction by any user. Zack's APIs provide a number of shared-state and inter-application communication mechanisms that support this high level of substitution, even for applications that need a high level of data persistence. This allows a multi-machine installation to scale linearly in the number of machines and also greatly simplifies load-balancing and fail-over configurations.

C. Performance Metrics

- HTML Pipelining methodologies to minimize latency
- Testing methodologies: server needed as backstop
- Zack Servers currently target the equivalent of 500 concurrent dialup modems per unit
- 3 to 5 Mbps, depending upon traffic composition
- 5,000 to 20,000 users, depending upon oversell ratio
- Typically 1,000 or more broadband connections per unit

1. Network performance metrics

There are several candidate metrics for measuring network performance of a Zack Server: bandwidth, transactions/objects-per-second and added latency.

Of these, bandwidth is the most useful measure for assessing the scalability of the system. Objects-per-second is a very tempting measure, because many cache vendors use it to measure their own products. Ultimately, however, the work done by a cache and by a Zack Server is completely different so the comparison provides little value. Latency is the most important of the three measures in terms of user-perception of network performance (see “HTML Pipelining” below).

2. User perceptions of network performance and HTML Pipelining

Pipelining, also called pipeline processing, is a technique used in advanced microprocessors where the microprocessor begins executing a second instruction before the first has been completed. That is, several instructions are in the pipeline simultaneously, each at a different processing stage.

In Zack’s “HTML Pipelining,” an Internet data stream is divided into segments and each segment can be processed concurrently with the other segments. When one application completes an operation on a segment, it passes the result to the next application in the pipeline and fetches the next segment from the preceding operation. The final results emerge at the end of the pipeline in rapid succession and are reassembled in the user’s Internet browser. HTML Pipelining is fundamental to the architecture of the Zack Server because it ensures that each transaction is being

processed as quickly as possible. Using this method, a normally loaded Zack Server typically adds less than 10 milliseconds to each chunk of response data, which means that users do not perceive any response difference between a Zack-enabled Internet session and a non-Zack-enabled session.

Informal studies done in the early stages of development of the Zack Server revealed that when Web browsing users are most sensitive to network/origin-server latency and less sensitive to actual network bandwidth. Typical page load times on a modern high-traffic Web site are quite long (1 to 4 seconds, measured from the user's click to the appearance of the primary content on the page). The threshold of user perception for added latency in such an environment is about 0.3 seconds (300 milliseconds).

Because Zack found latency to be the crucial factor in user perception of network performance, the design team set out to minimize latency in the operation of a Zack Server through the use of HTML Pipelining. Even a saturated Zack Server adds typically less than 50 milliseconds to a given data request, far below the threshold at which user will perceive the network to be any "slower".

3. Performance of shipping products

Shipping products are designed to support 3 to 5 Mbps (megabits-per-second) of real-world traffic per server unit, depending on the exact composition of the traffic. This corresponds to 500 active dialup-modem ports, which corresponds to 5,000 to 20,000 registered ISP users depending on oversell ratios.

Typical added latencies are in the range of 10 milliseconds for each chunk of response data (but often less), which is significantly below the level users will perceive.

V. Summary

Zack Systems has a several points of technological competitive differentiation. Zack's Servers are able to control access and administer applications, as well as integrate with profile, billing and administration interfaces at an ISP. These capabilities result from various components of the Zack Server that include: the Session Manager, Traffic Controller, Application Controller, the Zack APIs and the Active Marketing Toolkit.

A. Session Manager

The Zack Servers deliver the ability to identify specific traffic requests, allowing additional functionality to be delivered from the edge of the network on top of the Internet traffic. This feature enables ISPs to offer multiple classes of service across the entire user base without having to build complex network architecture. ISPs can also offer targeted and interstitial advertising and enhanced services that are personalized for each user.

B. Versatility of the Traffic Controller

The Traffic Controller is capable of acting as several different styles of network components including a proxy server, a traffic redirector (software layer 4 router), a content regulator, an application component server, a Web server and a cache. It is also able to integrate with existing network architecture, including caching infrastructures.

C. Capabilities of the Application Controller

The Application Controller enables an administrator at an ISP to segment its subscriber base according to application availability, subscription fee and advertising frequency. The ISP can also integrate the Zack Server directly with its billing system, enabling automatic billing integration.

D. Zack APIs

The Zack APIs allow an ISP to control the components of the Server architecture (the Traffic Controller, Application Controller and Session Manager) individually or en-masse. An SDK allows applications to be built above this core by Zack and third-party developers. The variety of capabilities and the ability to bring them to bear individually or in combination on a data stream is a distinct point of competitive differentiation.

E. Active Marketing Toolkit

The ability to automate the marketing of the enhanced services ensures a steady and consistent demand for applications, without the high cost of traditional marketing techniques. This toolkit enables free trials of applications and creates other types of instant marketing to reinforce the value of the applications.

F. Conclusion

In any network, the service provider who controls the edge of the network closest to the consumer has an inherent advantage. In television, this translates into a local affiliate's ability to deliver local ad content and programming. Similarly, in telephony the local operating company is able to profit from enhanced services such as caller ID, call waiting and voicemail. In the Internet, however, the companies in the best position to leverage the edge of the network (ISPs) have yet to do so and missed valuable opportunities to derive new revenue from enhanced services, targeted advertising and sponsorship opportunities.

Zack provides the means for ISPs to differentiate their services and drive up user retention as well as increase direct revenues and those from third parties. The Zack Servers deliver a broad suite of applications through an open, flexible API architecture, a persistent user interface and a high-performance proxy engine. In effect, Zack tilts the economics of the Internet back in favor of ISPs, giving them a unique opportunity to establish a more persistent, personalized and profitable relationship with their users.