IBP – The Internet Backplane Protocol: Two Page Summary [1]

Micah Beck, James S. Plank, Terry Moore

The Internet Backplane Protocol (IBP) is the result of thinking about storage in non-traditional ways. Heretofore, the combination of standard distributed file systems, stateless communication and caching has sufficed to deliver a good working environment for a variety of computing needs. However, given many of today's trends involving widely distributed, shared resources, there is a need for more flexible management of storage.

IBP differs from standard storage management systems (e.g. distributed file systems or databases) in three fundamental ways:

- It serves up writable storage (as opposed to merely readable storage) as a wide-area network resource.
- It allows for the remote direction of storage activities.
- It decouples the notion of user identification from storage.

We consider each of these in turn.

It serves up writable storage as a wide-area network resource

Currently, "the Internet" as we know it, is mainly a stateless communication substrate that serves up two kinds of network resources to its generic clients: read-only storage through anonymous FTP and the web, and remote processing servers that connect to clients via two-way ASCII communication pipes with Telnet. There are projects that are trying to enlarge this resource space, such as Jini, NetSolve, and active disk and network movement. IBP enlarges this resource space by focusing on storage, namely writable storage.

The benefits of offering writable storage as a network resource are numerous:

- Quality of service guarantees for networking can be met more easily when the intermediate routing nodes can store the communication buffers.
- Resource schedulers can include the staging of data near the processing resources for better resource utilization and better scheduling.
- Content services can be enhanced with both client and server-driven replication strategies (including, but not limited to caching, content push, multicast support, replica management) for improved performance.
- Fault-tolerance may be achieved.

Currently, most strategies for achieving the above benefits are *ad hoc* workarounds of the existing Internet architecture.

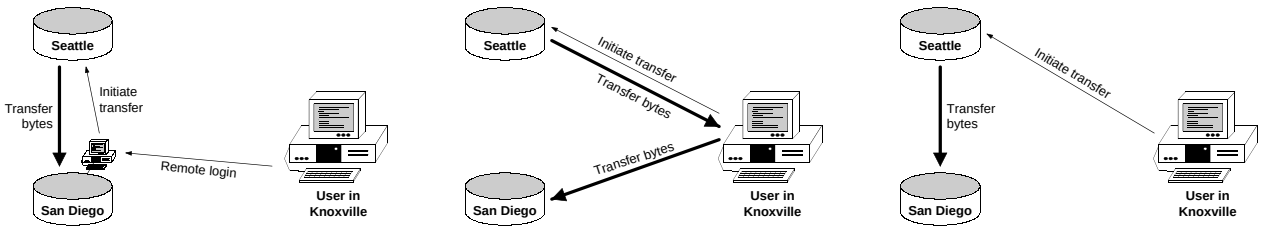
It allows for the remote direction of storage activities

Fundamentally, IBP is about the management of remote storage. This storage may be viewed as files or buffers, located on reliable stable storage, in RAM, or perhaps on an active disk. IBP allows a user or processing entity to both access and manage these storage entities *remotely*, without being involved in the actual manipulation of the bytes.

As an illustration, suppose a user in Knoxville wants to move a file from a machine in Seattle to a machine in San Diego. In today's world, there two ways that the user can do this. If he has a login account on the San Diego machine, he can log into this account from Knoxville, and then transfer the file (via FTP) from Seattle to San Diego as in Figure 1(a). Alternatively, if he is using a widely distributed file system on which he can mount disks from Seattle and San Diego, then he can perform a standard file copy from one disk to the other. Unfortunately, this requires that all the bytes of the file be shipped from Seatlle to Knoxville, and then from Knoxville to San Diego as in Figure 1(b).

With IBP, copying a file from one place to another is a basic operation that can be initiated remotely, without requiring the initiating user to have accounts on either machine. Thus, the transfer is as depicted in Figure 1(c), which is an optimal protocol.

[1] Technical Report CS-98-407, University of Tennessee, November, 1998. Contact information: `[mbeck,plank,tmoore]@cs.utk.edu`. The IBP web page is `http://www.cs.utk.edu/~plank/IBP`.



(a) With remote login and FTP (b) With a distributed file system (c) With IBP

Figure 1: Copying a file from one remote location to another.

It decouples the notion of user identification from storage

Systems for storage and networking usually differ in one fundamental way: storage systems typically have strict access control based on user id's, while networking systems usually offer as much bandwidth as they have to whoever currently wants it. The reason for this is sound. The use of networking resources is transient – once the bytes are sent, the resources are free. In contrast, when bytes are written to a storage system, they typically do not go away until the writer or a trusted agent explicitly deletes them.

IBP takes this concept one step futher. It offers up writable storage on the network to unauthenticated clients. That clients are unauthenticated does not mean that the system is chaotic or without safeguards. IBP allows the owner of a storage server to define how much storage to serve for IBP and how that storage should be served. In particular, IBP file allocation includes the notion that files may have a limited lifetime before they are removed by the IBP system. Alternatively, an IBP file may be allocated as *volatile*, meaning that the IBP server may revoke the storage at any time. Such a system strikes a balance between offering the benefits of writable storage on the network, and making sure that the owner of such storage has the ability to reclaim it when desired.

On the client end, IBP uses a capability-based naming system, in which IBP determines the name of each file, and the name is the only point of authentication between the client and server. This allows users on remote systems to pass IBP file names freely among each other without requiring them to pass through an authentication system that would otherwise be a bottleneck. When the IBP server removes a file, the capability is automatically invalidated.

The Backplane Metaphor

IBP derives its name from the fact that it enables a user to treat the Internet as its backplane. Whereas on a typical backplane, the user has access to memory and peripherals and can direct communication between them with DMA, IBP gives the user access to remote storage and standard Internet resources (content servers and others implemented with standard sockets) and can direct communication between them with the IBP API. With IBP, the catch phrase "the network is your computer" becomes more of a reality.

Current State of Development

The IBP API has been developed and is available at `http://www.cs.utk.edu/~plank/IBP/API`. It is currently implemented over TCP/IP sockets that make use of server daemons invoked by the storage resource owners, and a library of client calls to be linked with client applications. Within the next few months, the Internet2 Distributed Storage Infrastructure program will be deploying between 6 and 10 mass storage systems in various locations on the Internet. These will be running IBP servers to serve a subset of their resources to IBP clients.

In the application domain, we have developed a *logistical FTP server* using IBP that allows a FTP client to schedule a FTP-get sometime in the future. If the FTP server is IBP-enabled, it then moves the relevant file to an IBP server close to the client (perhaps even on the client), so that when the client performs the FTP-get, it occurs with better and more reliable bandwidth. This also allows a *logistical FTP scheduler* to schedule reservations using intermediate staging so that all reservations may be met. This leads to much better performance and use of resources than the current state of FTP.

We are currently developing IBP-enabled clients and servers for the NetSolve network computing system. IBP allows NetSolve to allocate and schedule storage resources as part of its resource brokering. Again, this will lead to much improved performance, and enables fault-tolerance when resources fail or are revoked.