Exhibit B-60

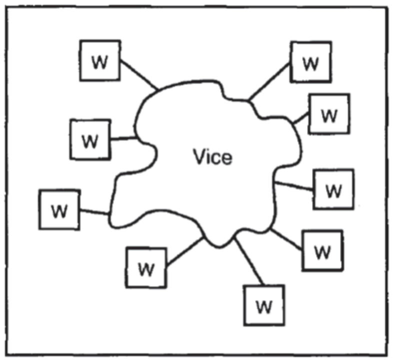
Invalidity Claim Chart for U.S. Patent No. 6,415,280: Satyanarayanan, M., *Scalable, Secure, and Highly Available Distributed File Access, IEEE Computer*, vol. 23, no. 5 (May 1990), pp. 9–21

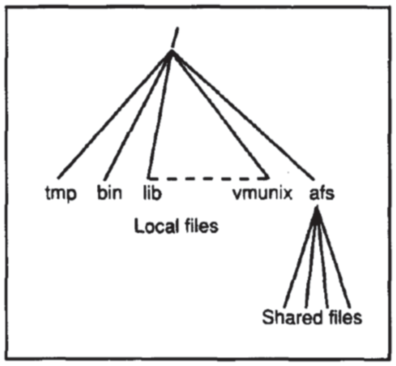
Satyanarayanan, M., *Scalable, Secure, and Highly Available Distributed File Access*, IEEE Computer, vol. 23, no. 5 (May 1990), pp. 9–21 ("*Satyanarayanan II*") is available as prior art at least under 35 U.S.C. § 102(b). Langer, A., "Re: dl/describe (File descriptions)," article <1991Aug7.225159.786@newshost.anu.edu.au> in Usenet newsgroups "alt.sources.d" and "comp.archives.admin" (August 7, 1991) ("*Langer*") is available as prior art at least under 35 U.S.C. § 102(b). Kantor, F.W., "FWKCS Contents-Signature System Version 1.22," Aug. 10, 1993 ("*Kantor*") is available as prior art at least under 35 U.S.C. § 102(b).

To the extent it is found that *Satyanarayanan II* does not expressly disclose certain limitations, such limitations are inherent. Moreover, to the extent it is found that *Satyanarayanan II* does not anticipate any asserted claim, *Satyanarayanan II* renders it obvious, either alone or in combination with the knowledge of a person of ordinary skill in the art, and/or in combination with other prior art references identified in the cover pleading or herein.

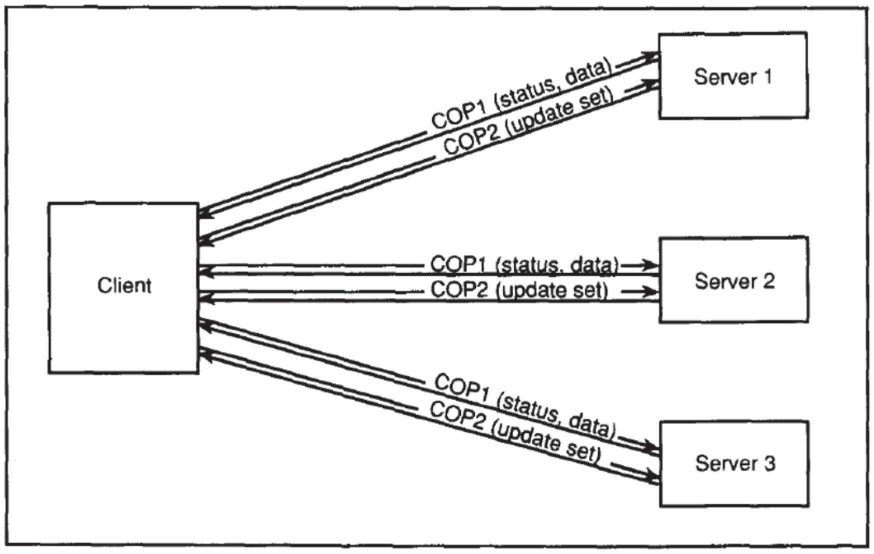
The cited portions of the prior art references are only examples, and Defendants reserve the right to rely on any further uncited portions of the prior art references as additional evidence that the references disclose and/or render obvious a claim limitation. Defendants' Invalidity Contentions are not an admission that the Accused Instrumentalities infringe the asserted claims of the '280 patent.

Claim 36	*Satyanarayanan II*
[a] A method of delivering a data file in a network comprising a plurality of processors, some of the processors being servers and some of the processors being clients, the method comprising:	*Satyanarayanan II* discloses "a method of delivering a data file in a network comprising a plurality of processors, some of the processors being servers and some of the processors being clients." For example, *Satyanarayanan II* discloses the Coda file system, which is based on the Andrew File System (AFS) architecture. AFS comprises a plurality of processors, including file servers and workstations (clients). The system delivers data files to clients through a standard Unix file system interface.

Claim 36	***Satyanarayanan II***
	 "Figure 1. A high-level view of the Andrew architecture. The structure labeled 'Vice' is a collection of trusted file servers and untrusted networks. The nodes labeled 'W' are private or public workstations, or timesharing systems. Software in each such node makes the shared files in Vice appear as an integral part of that node's file system." (*Satyanarayanan II* at 10.) "Data sharing in Andrew is supported by a distributed file system that appears as a single large subtree of the local file system on each workstation. The only files outside the shared subtree are temporary files and files essential for workstation initialization. A process called Venus, running on each workstation, mediates shared file access. Venus finds files in Vice, caches them locally, and performs emulation of Unix file system semantics." (*Satyanarayanan II* at 10.)

Claim 36	*Satyanarayanan II*
	 / tmp bin lib vmunix afs Local files Shared files “Figure 2. File system view at a workstation: how the shared files in Vice appear to a user. The subtree under the directory labeled ‘afs’ is identical at all workstations. The other directories are local to each workstation. Symbolic links can be used to make local directories correspond to directories in Vice.” (*Satyanarayanan II* at 10.) “Coda provides a scalable and highly available approximation of Unix semantics. . . . In the absence of failures, Coda and AFS-2 semantics are identical.” (*Satyanarayanan II* at 16.)
[b] storing the data file is on a first server in the network and storing copies of the data file on a set of servers in the network distinct from the first server; and	*Satyanarayanan II* discloses “storing the data file is on a first server in the network and storing copies of the data file on a set of servers in the network distinct from the first server.” For example, when a client modifies a data file on the Coda file system, the system stores the data file on a first server (“preferred server,” or PS) and also stores copies of the data file on a set of servers (other servers within the “accessible volume storage group,” or AVSG) in the network, distinct from the first server. *Satyanarayanan II* replicates volumes on a plurality of servers (at least three), and thus

Claim 36	***Satyanarayanan II***
	stores copies of files on a plurality of servers. "The unit of replication in Coda is a volume. A *replicated volume* consists of several physical volumes, or replicas, that are managed as one logical volume by the system. Individual replicas are not normally visible to users. The set of servers with replicas of a volume constitutes its *volume storage group* (VSG). . . . For every volume from which it has cached data, Venus keeps track of the subset of the VSG that is currently accessible. This subset is called the *accessible VSG* (AVSG)." (*Satyanarayanan II* at 16.) "When servicing a cache miss, Venus obtains data from one member of its AVSG, known as the *preferred server*. The PS can be chosen at random or on the basis of performance criteria such as physical proximity, server load, or server CPU power." (*Satyanarayanan II* at 16.) "When a file is closed after modification, it is transferred to all members of the AVSG. This approach is simple to implement and maximizes the probability that every replication site has current data at all times. Server CPU load is minimized because the burden of data propagation is on the client rather than the servers. This in turn improves scalability, since the server CPU is the bottleneck in many distributed file systems." (*Satyanarayanan II* at 17.) "Figure 6 illustrates the message exchange in a store operation (which corresponds to a file close)." (*Satyanarayanan II* at 17.)

Claim 36	*Satyanarayanan II*
	 Client Server 1 Server 2 Server 3 COP1 (status, data) COP2 (update set) COP1 (status, data) COP2 (update set) COP1 (status, data) COP2 (update set) “Figure 6. A store operation in Coda: the two phases of the Coda update protocol. In the first phase, COP1, the three servers are sent new status and data in parallel. In the later asynchronous phase, COP2, the update set is sent to these servers. COP2 also occurs in parallel and can be piggybacked on the next COP1 to these servers.” (*Satyanarayanan II* at 16.) *See also* Applicants Admitted Prior Art: “In some data processing systems in which several processors are connected in a network, one system is designated as a cache server to maintain master copies of data items, and other systems are designated as cache clients to copy local copies of the master data items into a local cache on an as-

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS

Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS

Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS

Sync your system to PACER to automate legal marketing.

WHAT WILL YOU BUILD? | **sales@docketalarm.com** | **1-866-77-FASTCASE**

fastcase® Smarter legal research.