# High-Speed Wide-Area Distributed Data Handling: Architecture and Implementation[1]

*William E. Johnston[2], William Greiman, Brian Tierney, Craig Tull*
*Information and Computing Sciences Division*
*Douglas Olson, Nuclear Science*
*Ernest Orlando Lawrence Berkeley National Laboratory*
*University of California*

1. This work is supported by the Director, Office of Energy Research, Office of Computation and Technology Research, Mathematical, Information, and Computational Sciences Division, of the U. S. Department of Energy under Contract No. DE-AC03-76SF00098 with the University of California, and by ARPA ISTO

2. wejohnston@lbl.gov, 510-486-5014, http://www-itg.lbl.gov/~johnston

# Data-Intensive Computing in Widely Distributed Environments

**The overall goal of this work is to provide methodology and tools to permit the scientific community to routinely deal with massive volumes of data at high data rates with complete location and access transparency.**

## Challenge

**Identify what must be done to produce**

- **predictable, high-speed, components**
- **that will compose to yield high-performance widely distributed applications .....**
  **rather than having to "tune" these systems from top-to-bottom as we mostly have to do now**

**Increasingly, we believe that meeting this challenge will involve**

- **comprehensive and adaptable monitoring that drives adaptation of the system components, and**
- **automated remote management**

## Issues and Approaches

- **Scalability - the system must grow and shrink “gracefully”**
  - **dynamic configuration**
- **Performance**
  - **system-level parallelism and pipelining**
  - **individual threads of control for every server-level resource**
  - **monitoring**
- **Reliability - otherwise the system is not real**
  - **autonomous monitoring and management ( + configurability and adaptability)**
- **Security - otherwise the system is not real**
  - **certificate based distributed management of policy**

# An Overall Model for Data-Intensive Computing

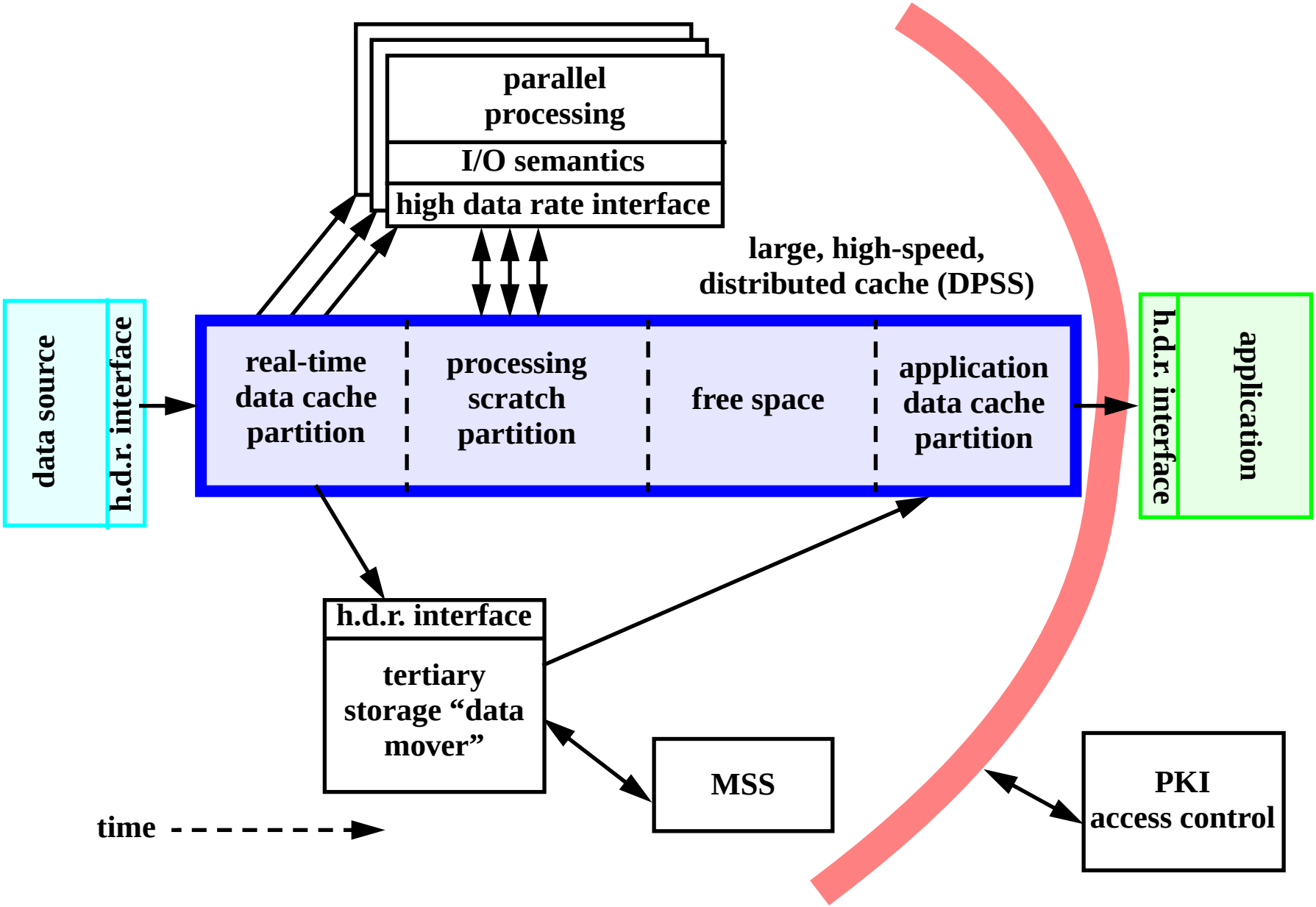
- Each data source deposits its data in a distributed cache, and each data consumer takes data from the cache, usually writing processed data back to the cache
- A standard interface to a large, high-speed, application-oriented cache
- In almost every case there is also a tertiary storage system manager that migrates data to and from the cache
- Depending on the size of the cache relative to the objects of interest, the storage system manager may move objects to the cache over some relatively long period of time; that is, the cache can serve as a moving window on the object/dataset.

# Model

- **The cache - application interface is at the logical block level, but client-side libraries implement various access semantics (including C I/O):**
  - **upon request available data is returned,**
  - **requests for data in the dataset, but not yet migrated to cache, causes the application-level read to block.**

**Generally, the cache is large compared to the available disks of the computing environment, and very large compared to any single disk (e.g. hundreds of gigabytes).**

# Model

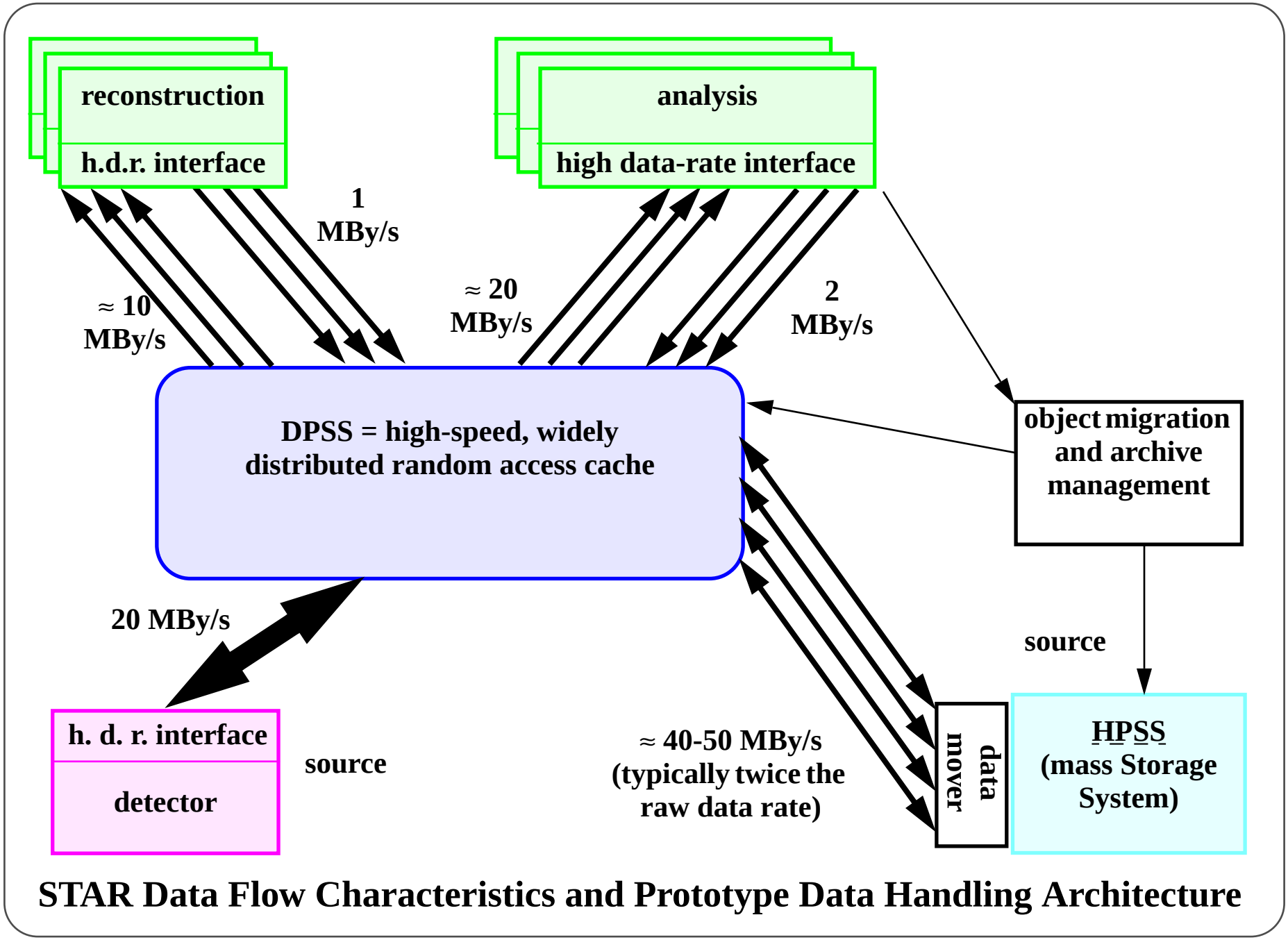



**Overall Data Acquisition and Processing Model**

# Example: Prototype Architecture for HENP Distributed Analysis

- **High Energy and Nuclear physics accelerator detectors are the prototypical high data-rate scientific instrument**
- **The WALDO system provides another example, in which a digital library is inserted in the architecture (see http://www-itg.lbl.gov/WALDO)**

# HENP Distributed Analysis

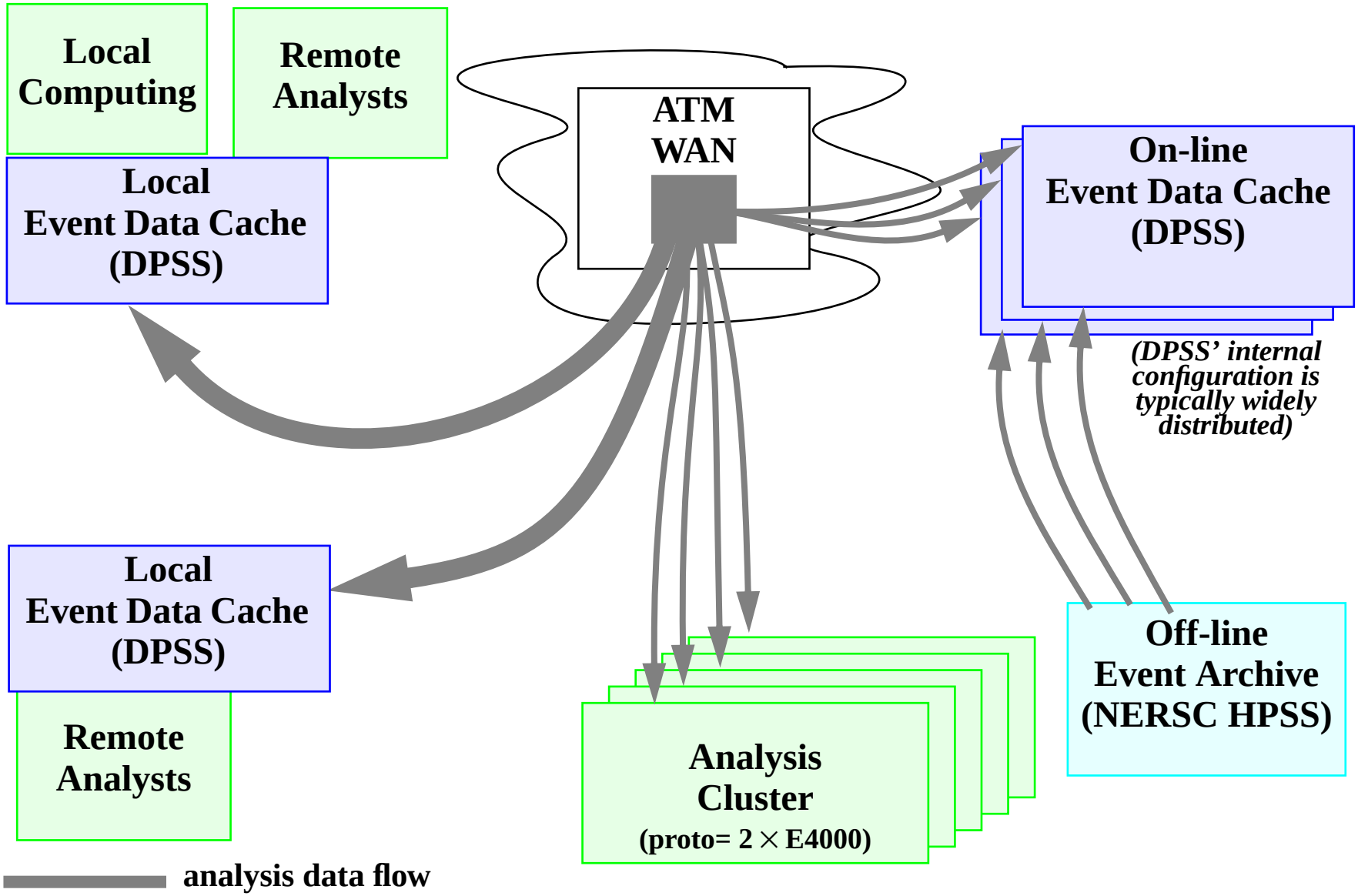



**STAR Data Flow Characteristics and Prototype Data Handling Architecture**

# HENP Distributed Analysis

- **The model for analysis is an off-line event archive feeding a high-speed cache, with remote analysts using a combination of local and remote distributed resources (figure 3.).**

# HENP Distributed Analysis




## Distributed Analysis Model

# The Distributed Cache Architecture

**The Distributed-Parallel Storage System (DPSS) is a high-speed, application-oriented network data cache that is itself a widely distributed system, and that serves several roles in high-performance, data-intensive computing environments.**

**Functionally, the DPSS provides:**

- **A standard interface for high-speed data access with the functionality of a single, very large, random access, block oriented I/O device (i.e. a "virtual disk");**
- **High capacity, on-line cache storage (we anticipate a terabyte size for the STAR analysis environment) that serves to isolate the application from the tertiary storage system and the instrument (detector data acquisition system);**

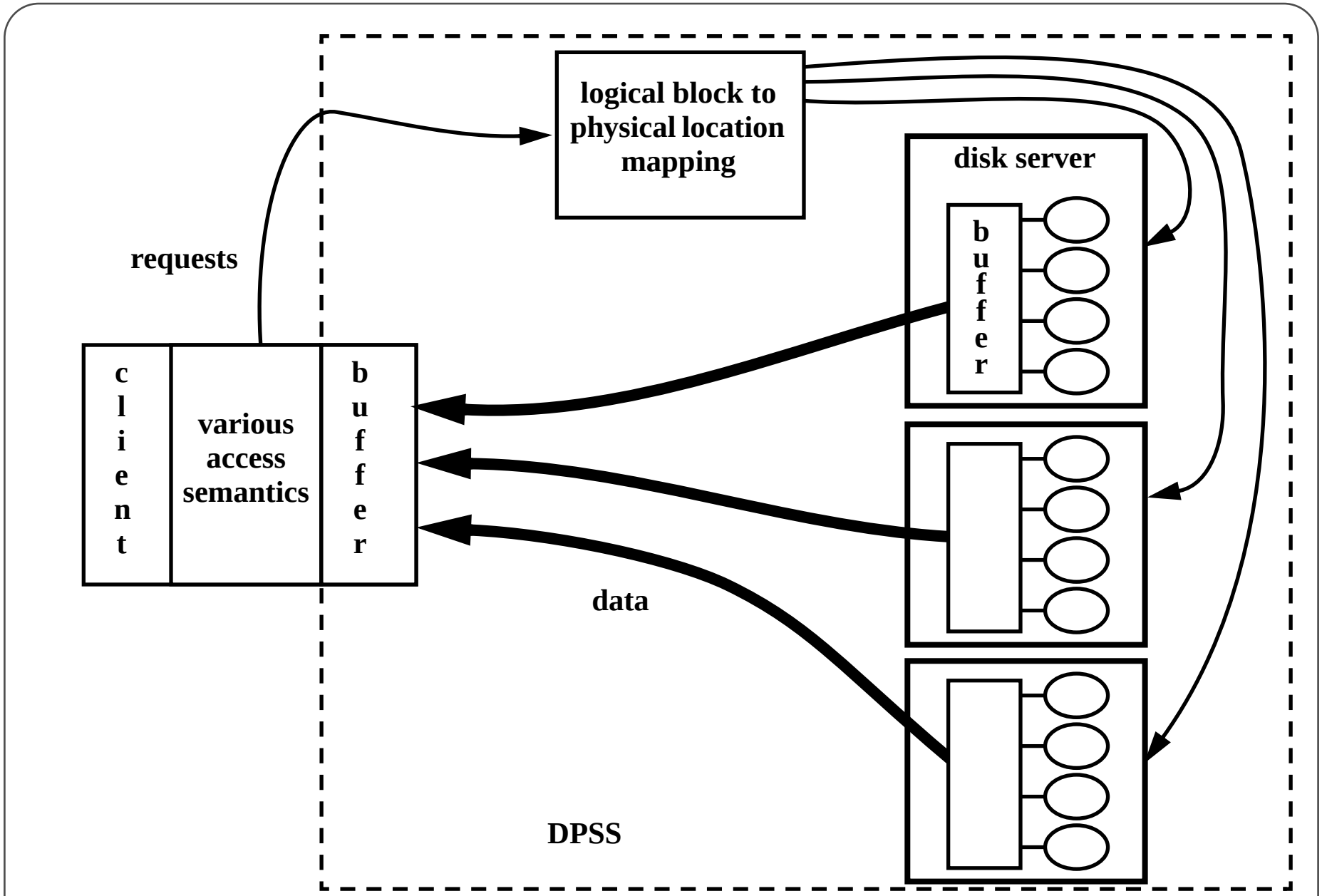
# Cache Architecture

- Access to many large datasets that may be logically present in the cache by virtue of the block index maps being loaded, even if the data is not yet available (in this way processing can begin as soon as the first data has been migrated from tertiary storage);
- Application-specific interfaces to an extremely large space (16 byte indices) of logical blocks;
- The ability to dynamically configure on-line systems by aggregating workstations and disks from all over the network (this is routinely done in the MAGIC testbed);
- The ability to build large, high-performance storage systems from the least expensive commodity components;
- Scalable performance by increasing the number of parallel operating DPSS servers.

# Cache Architecture

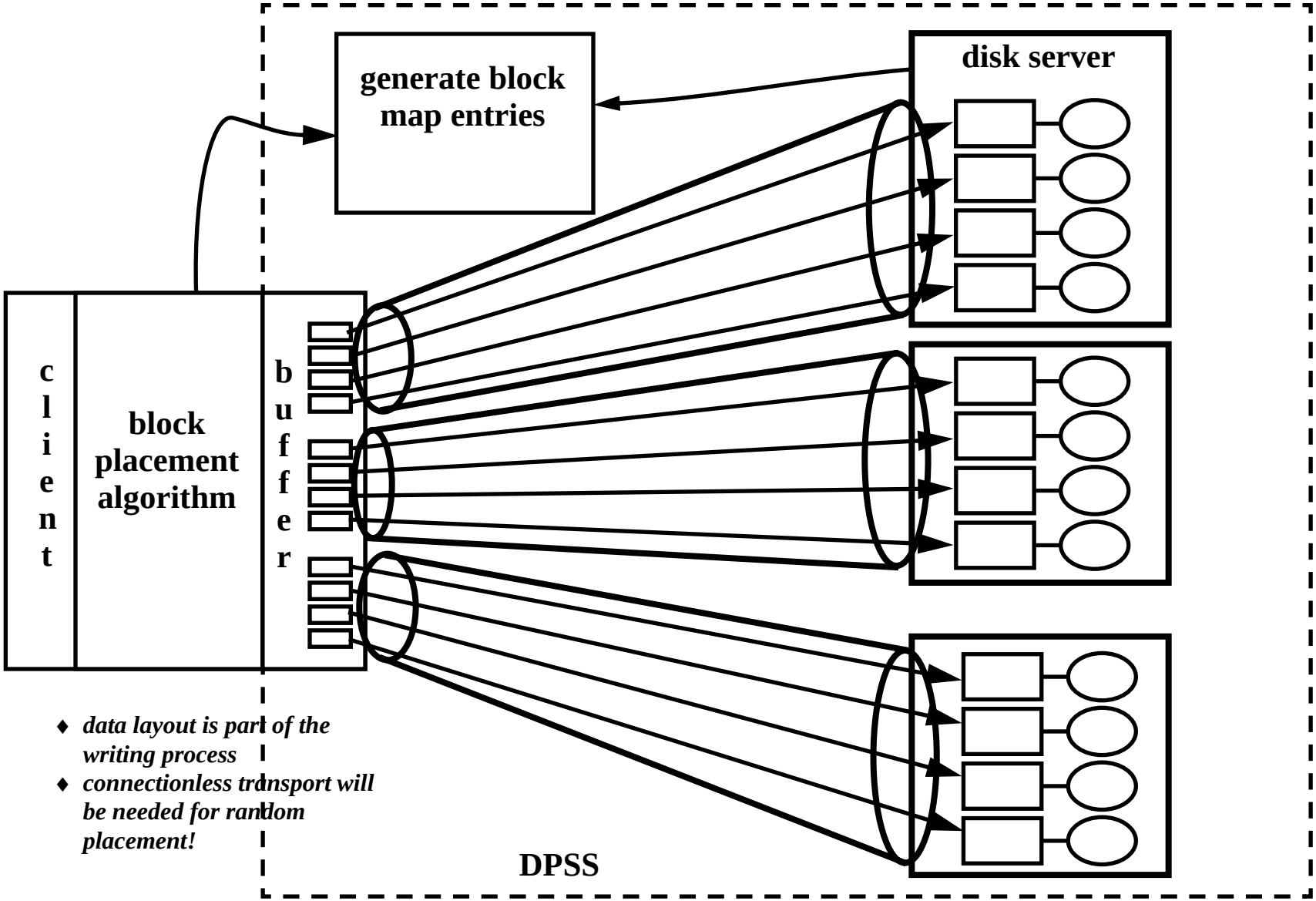
- DPSS uses parallel operation of distributed servers to supply, for example, image streams fast enough to enable various multi-user, “real-time”, virtual reality-like applications in an Internet / ATM environment.
- Ultimately, the ene-to-end performance comes from using separate threads of control for every resource
- As illustrated in figure 5. the DPSS provides a “logical block” server that does “third party” transfers from disk servers directly to application client buffers;
- The DPSS, as a system, is designed to be distributed across a wide-area network;
- The components are actively managed by a collection of independently communicating agents to provide highly distributed, reliable, wide-area operation;

# Cache Architecture




**Distributed-Parallel Storage System Model (Reading)**

# Cache Architecture



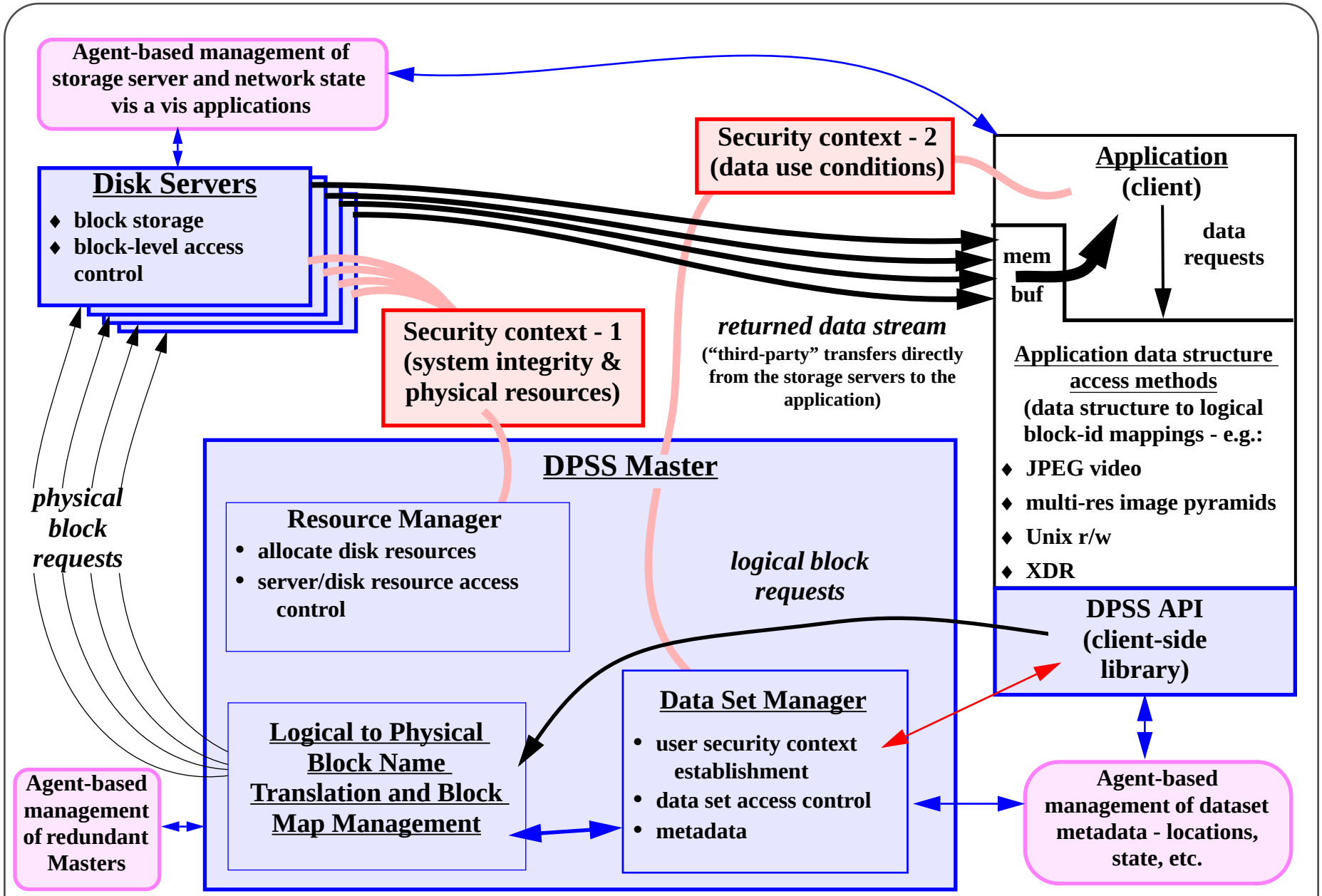

- *data layout is part of the writing process*
- *connectionless transport will be needed for random placement!*

## DPSS Model for High-Speed Writing

# Cache Architecture

- **Data placement occurs during the data-write operation (figure 8.)**
- **Three major architectural components: storage and control, security, agent-based management (figure 7.)**

# Cache Architecture




**Distributed-Parallel Storage System Architecture (data reading illustrated)**

# Performance Monitoring and Analysis

**There are virtually no behavioral aspects of widely distributed applications that can be taken for granted - they are fundamentally different from LAN based distributed applications.**

- **A hard problem that is a barrier to the routine construction of high-speed distributed applications**
- **Hard, because techniques that work in the LAN frequently do not work in the wide area**

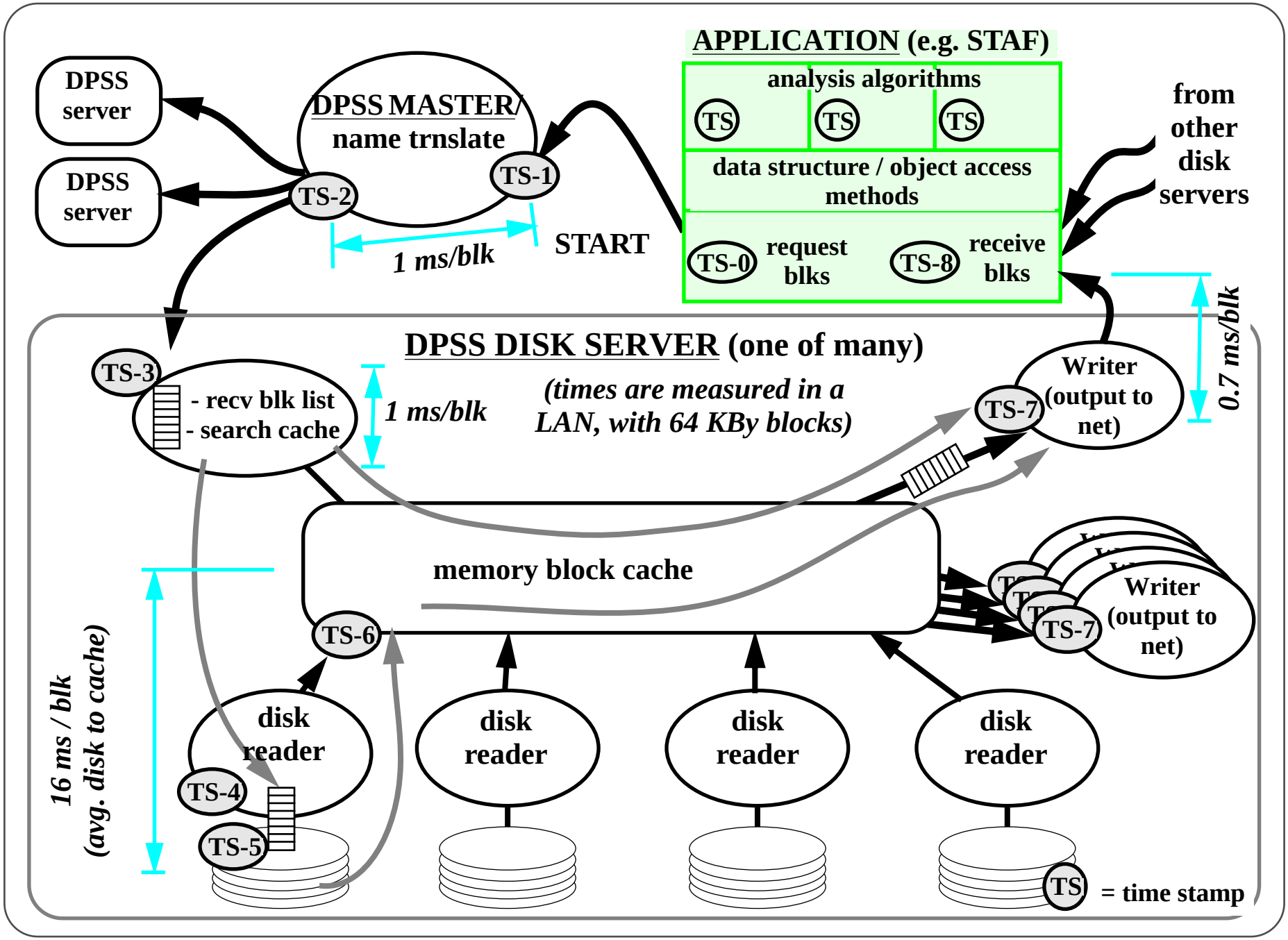
**To characterize the wide area environment we have developed a methodology for detailed, *end-to-end, top-to-bottom monitoring* and analysis of every significant event involved in distributed systems data interchange.**

- **Has proven invaluable for isolating and correcting performance bottlenecks, and even for debugging distributed parallel code**

# Performance Monitoring and Analysis

- **The monitoring methodology involves precision time correlation of events throughout the distributed system, together with analysis techniques for obtaining information from the reconstructed dataflow lifelines.**
  - **NetLogger/LogTracer tools collect state information and time stamps at all critical points in the data path, including instrumenting the clients and applications**
  - **timing and event information is carried as a defined part of the data block structure OR is logged and correlated based on the timestamps**
  - **NTP is used to synchronize system clocks to within about 250 microseconds of each other (but the systems have to stay up for a significant length of time for the clocks to converge to 250 μs)**
- **The results are detailed, data block transit history "life-lines"**

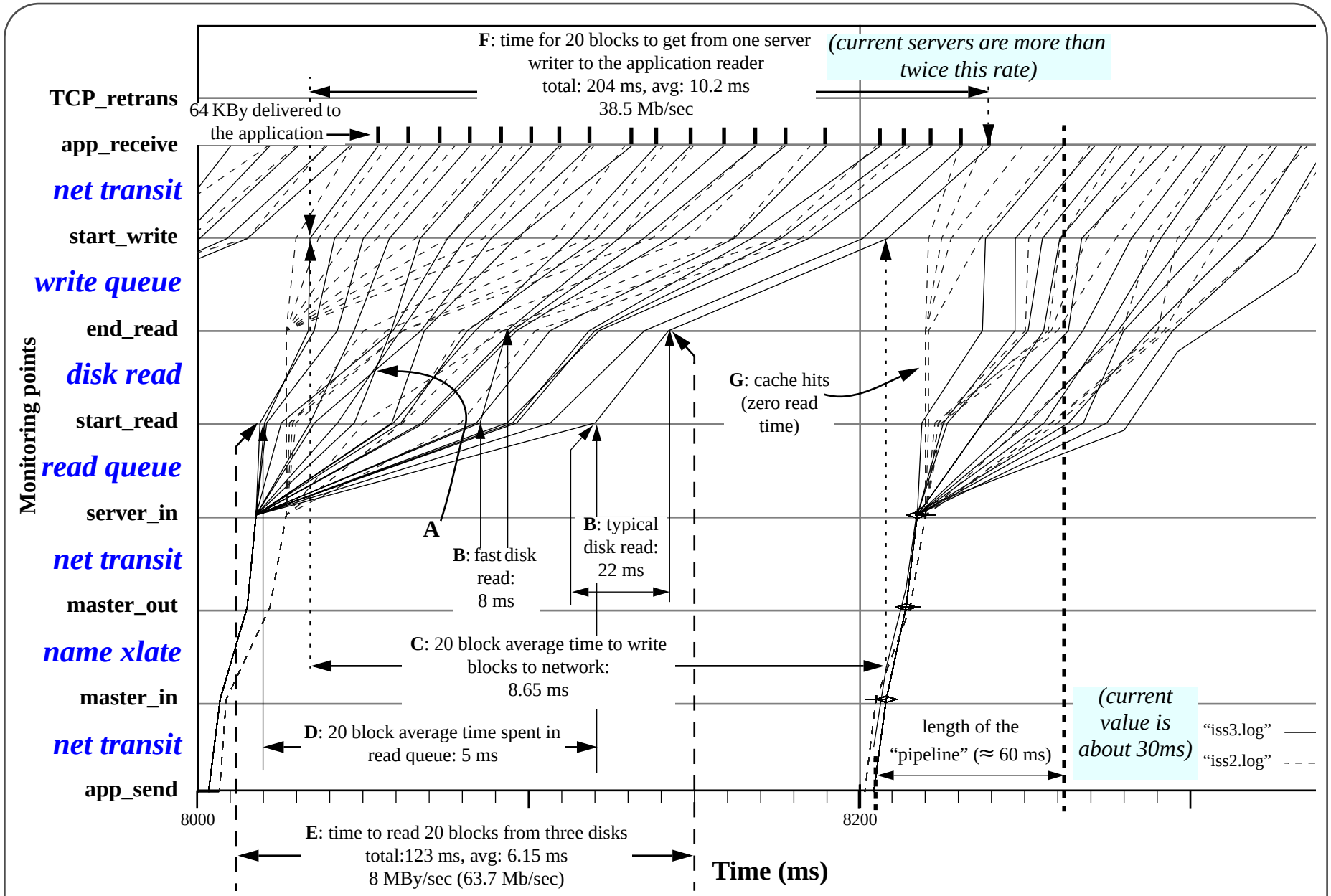
# Performance Monitoring and Analysis



# Performance Monitoring and Analysis

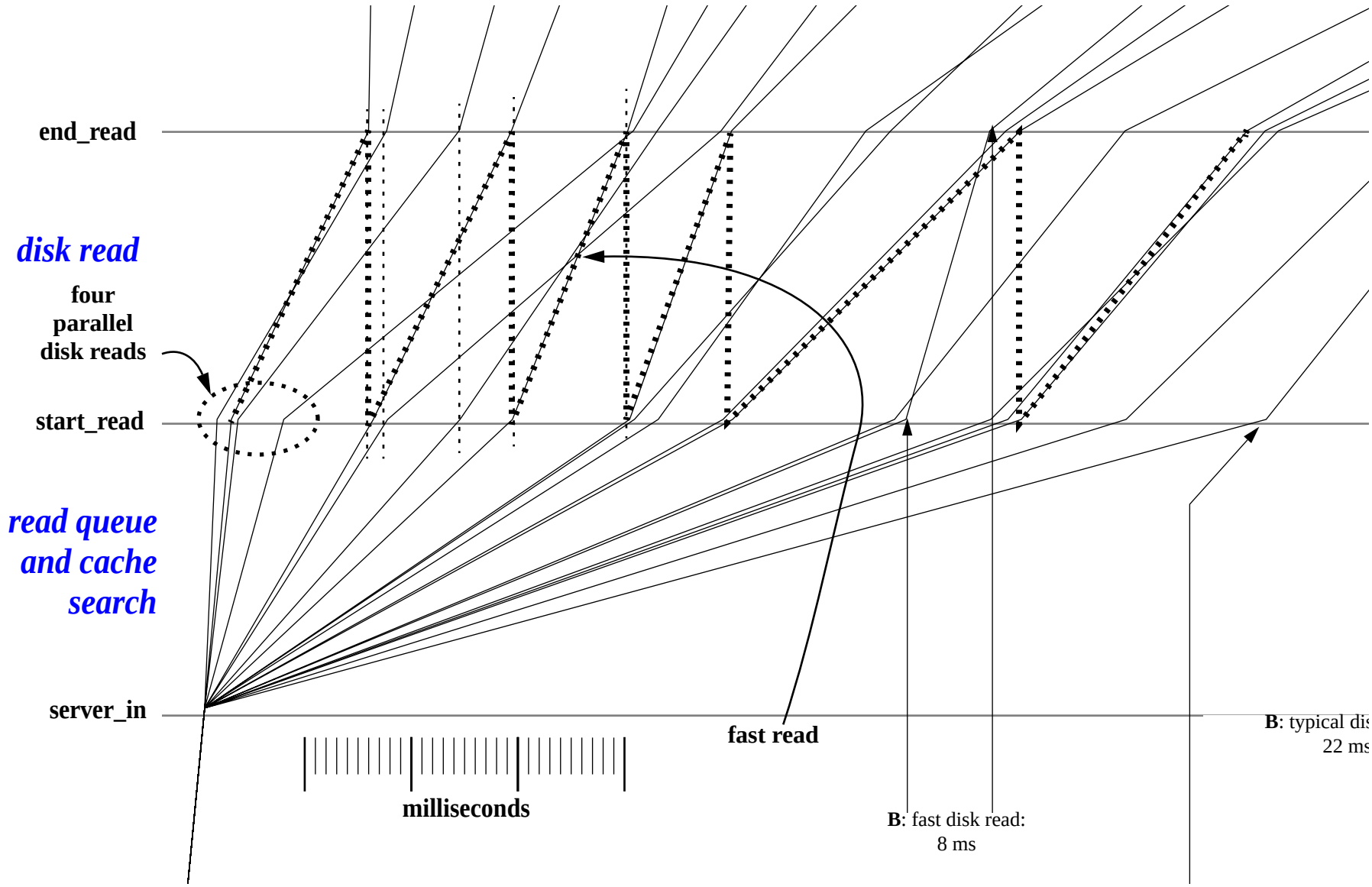
- **Analysis of the event life-lines**

# Performance Monitoring and Analysis




## Two server, ATM *LAN* Event lifelines

# Performance Monitoring and Analysis



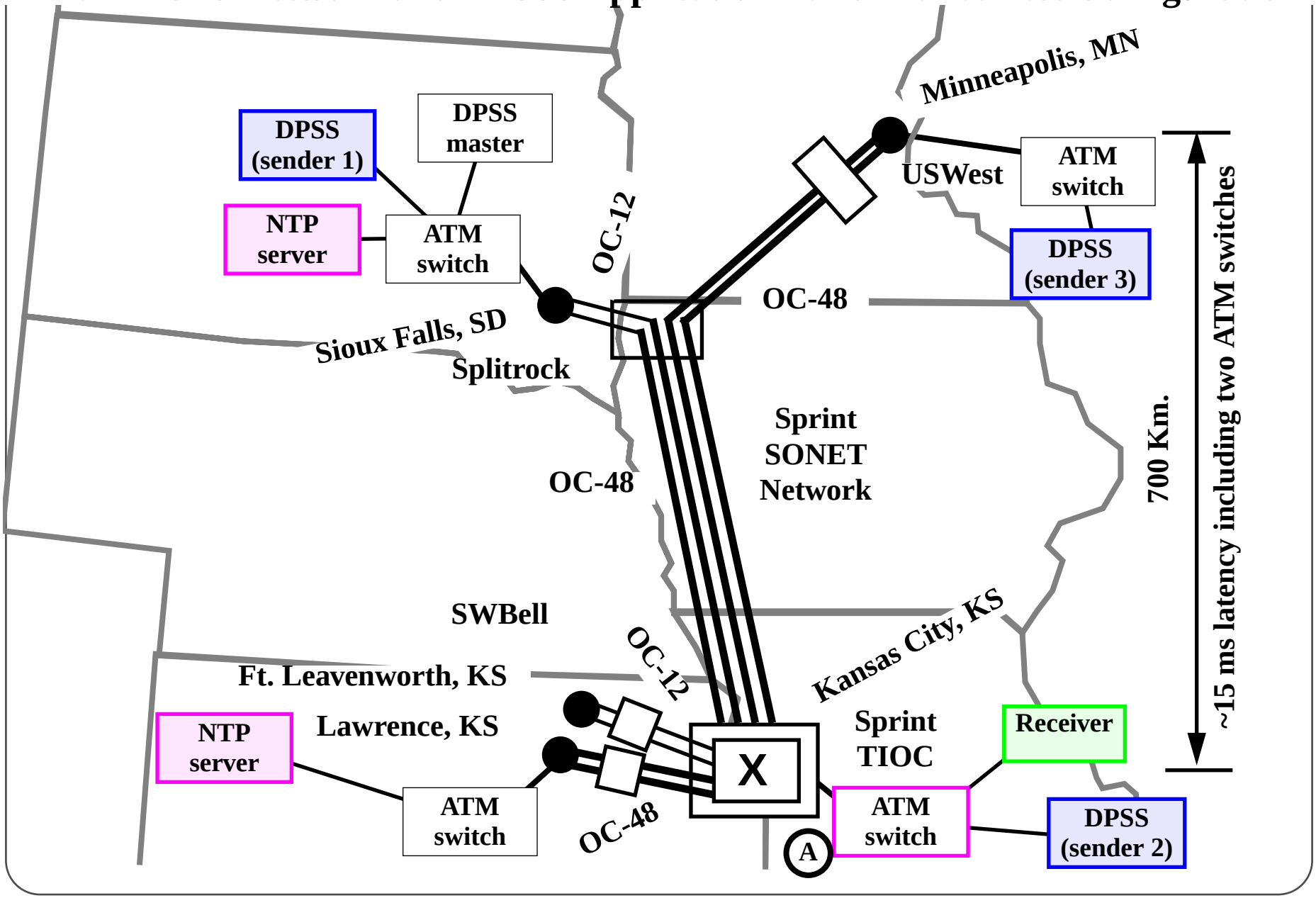

## Two server, ATM *LAN* Event lifelines

## Results of the MAGIC WAN experiment

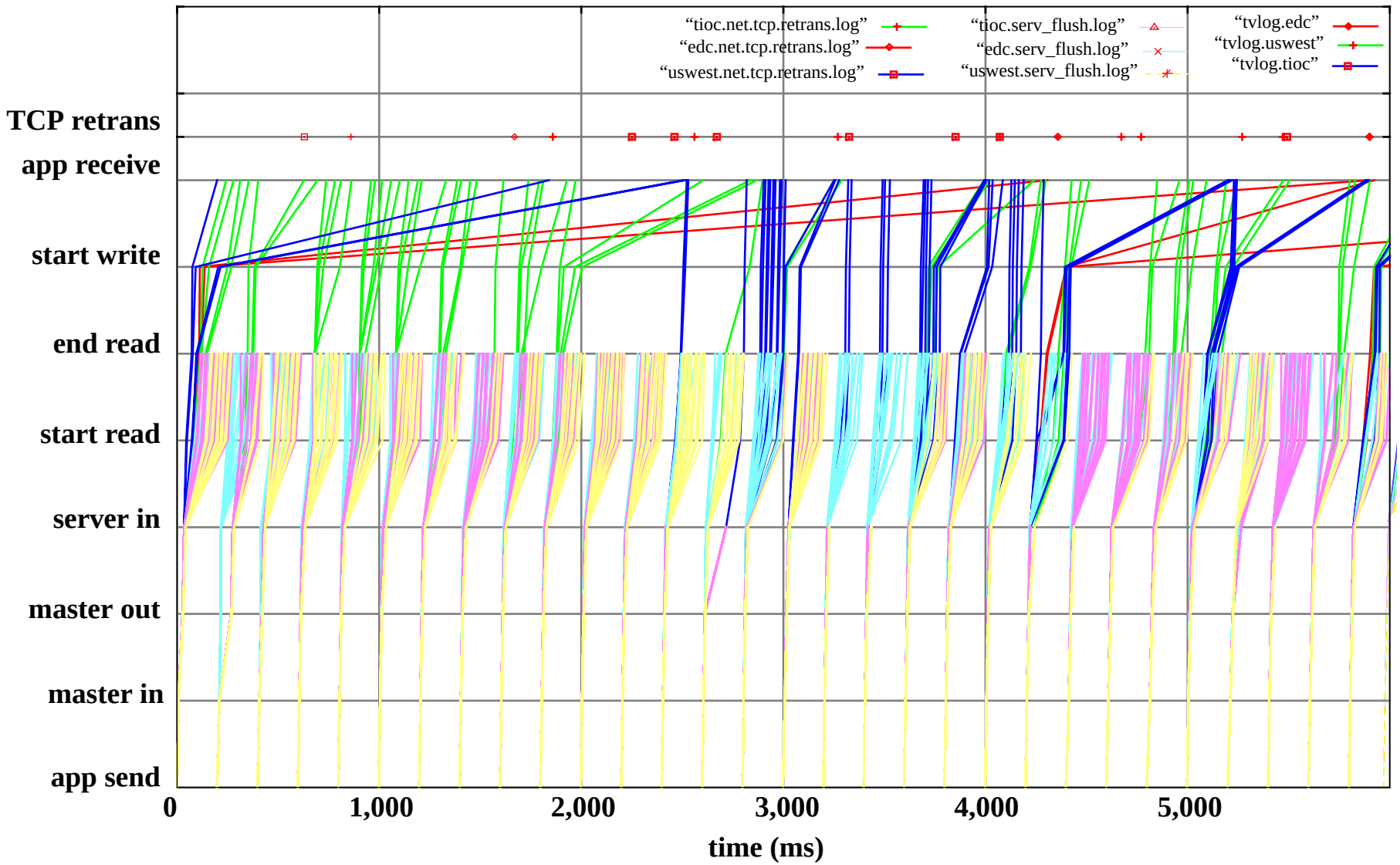
- Three disk server configuration (next figure) DPSS
- Things to notice:
  - Many TCP retransmissions, and some very long delays (up to 5500 ms!) (Once a block is written to the TCP socket, the user level flushes have no effect, and TCP will re-send the block until transmission is successful, even though the data is likely no longer needed and is holding up newer data).
  - These long delays are almost always accompanied by one or more TCP retransmit events.

# Performance Monitoring and Analysis

## The MAGIC Network and DPSS / Application Performance Test Configuration



# Performance Monitoring and Analysis

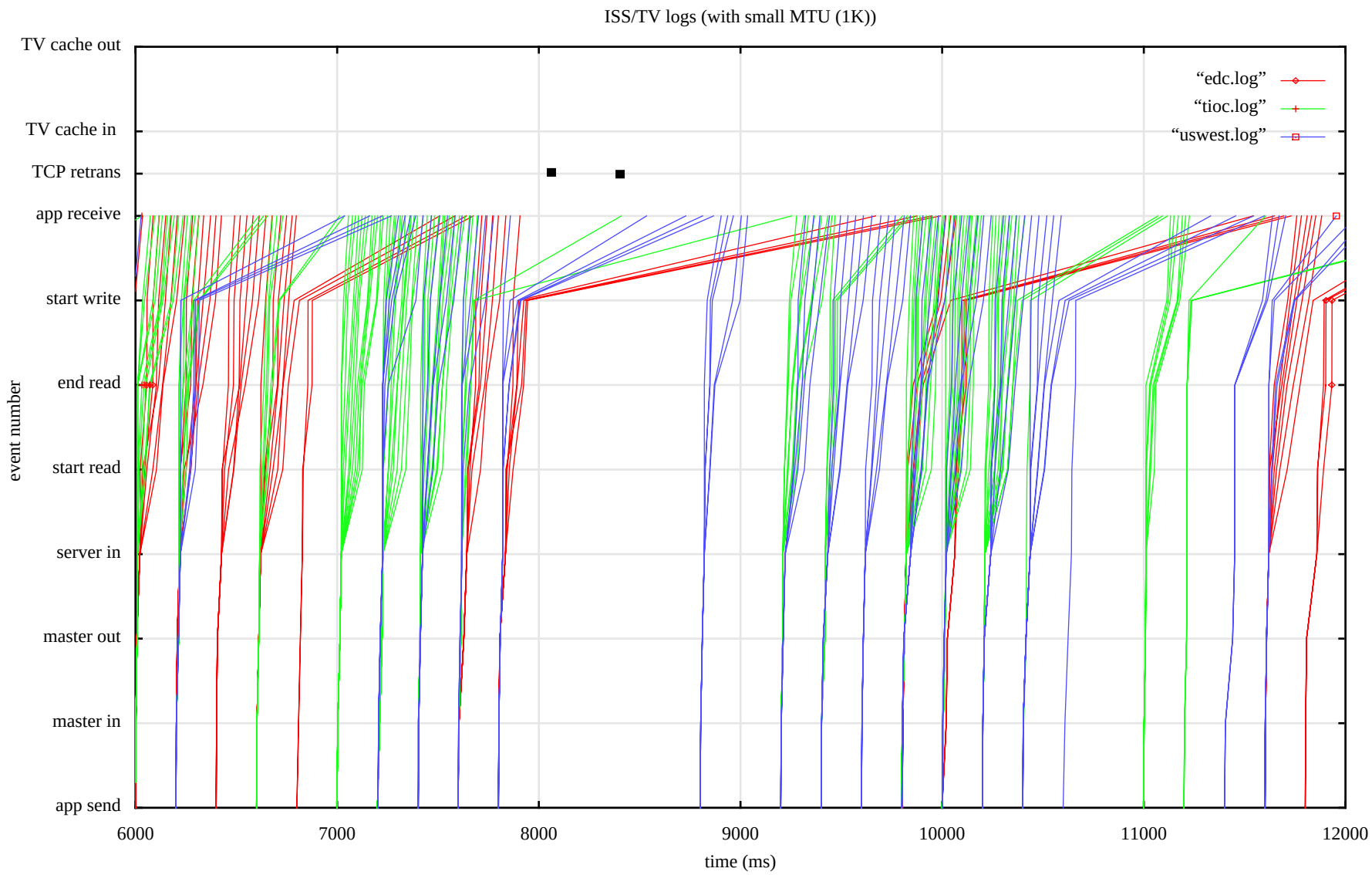



**Three servers, ATM WAN: things can go very wrong**

## Experiment to test TCP window vs. MTU hypothesis: Reduce MTU size

- Reduce the network MTU to a very small value (e.g. 1024 bytes) so that the TCP window can close to values more consistent with the available switch buffering
  - indeed, this helps a lot (see next figure)

# Performance Monitoring and Analysis



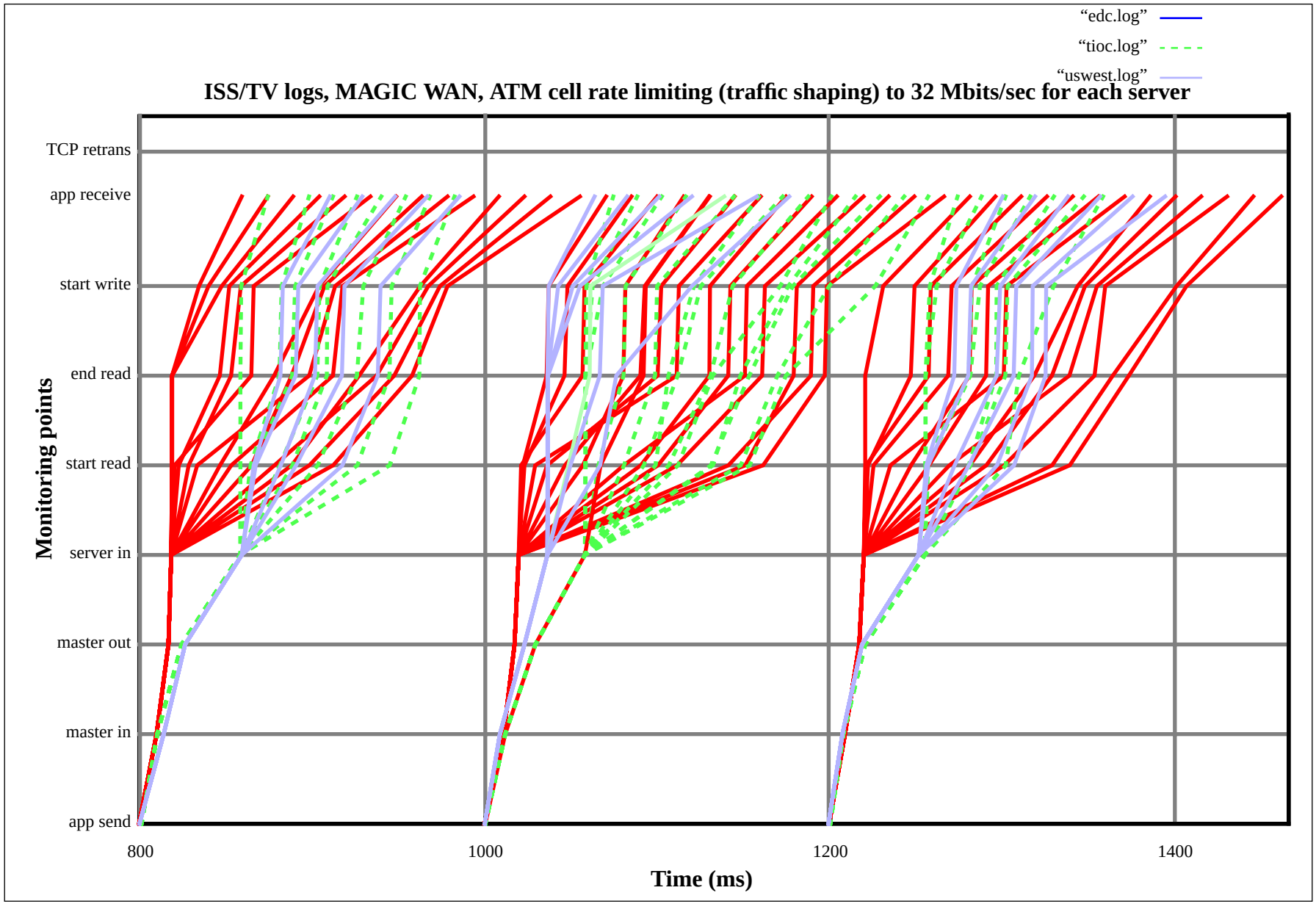

**Small MTU experiment, MAGIC WAN**

# Performance Monitoring and Analysis

## Experiment to test switch cell dropping hypothesis: Cell pacing

**These experiments caused all of the MAGIC backbone switches to be replaced and/or upgraded with large output buffers.**

# Performance Monitoring and Analysis



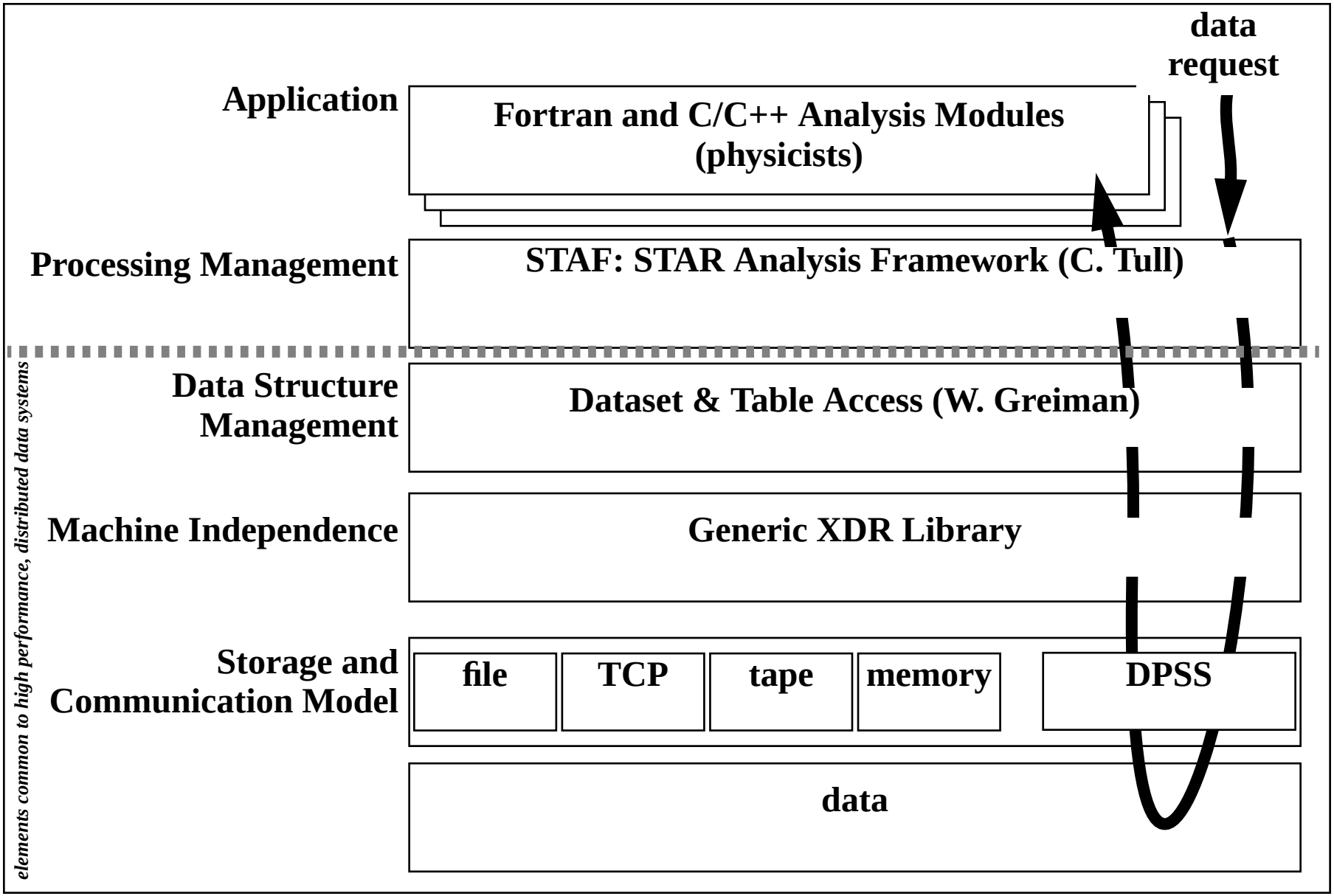

**The Three Server ATM WAN Experiment with Cell Rate Limiting**

# Application Performance Monitoring Example

**The STAR analysis framework (STAF) is being used to provide a realistic application environment in which to validate and refine the data handling architecture and implementation.**

**Generally speaking, STAF manages self-describing data structures on behalf of analysis modules. Data is requested through a standard interface that supports several communications models, including the DPSS cache. The data is converted to machine-specific format and placed into memory data structures, whence it is accessed by the analysis modules.**

# Performance

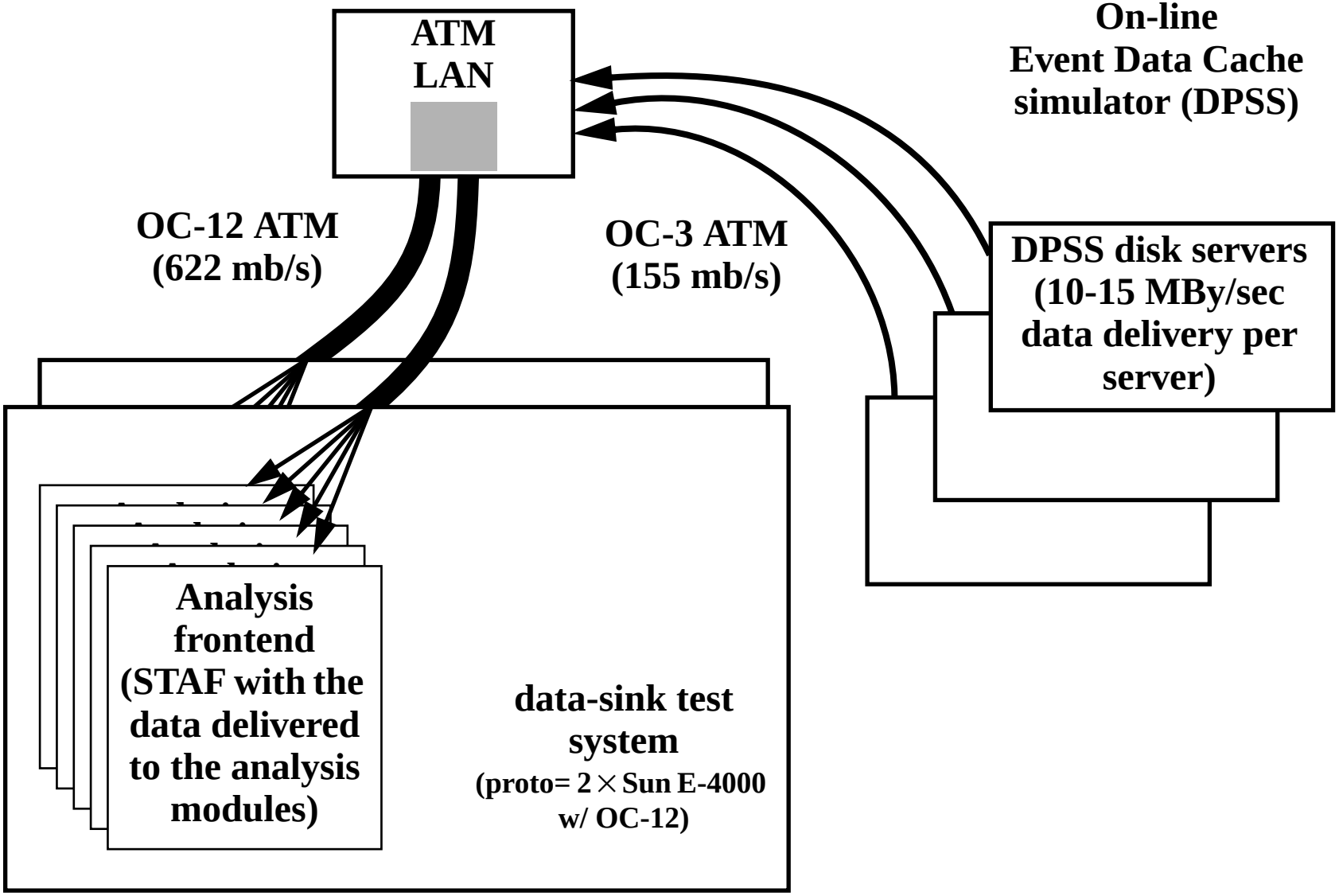



Software Architecture for Analysis (e.g. STAR)

# Performance

- A typical DPSS server consists of a commodity workstation (e.g. a 200 MHz Pentium) with one high speed network interface (100 Mb/s Ethernet or 155 Mb/s ATM), three or more SCSI adaptors, and three or more disks on each SCSI string.
- Each such server can independently deliver about 10 Mbytes/sec of data to a remote application which sees the aggregated streams for all servers in a DPSS system.
- Performance in an HENP-like configuration for multiple, parallel applications was measured using a two disk server, four disk, DPSS configuration. Data requests are made by a STAF based reader through the DPSS file semantics interface which collects blocks from the DPSS servers, buffers them, and provides serial access to the buffer through an API.

# Performance




**DPSS Performance Testing with Many Parallel Applications**

# Performance

***The throughput rates are measured as data is delivered to the application level (analysis modules),***
**a path that includes translating the data to the appropriate machine format and structuring it in memory (both of which are very fast operations), which provides the most realistic performance measurement.**

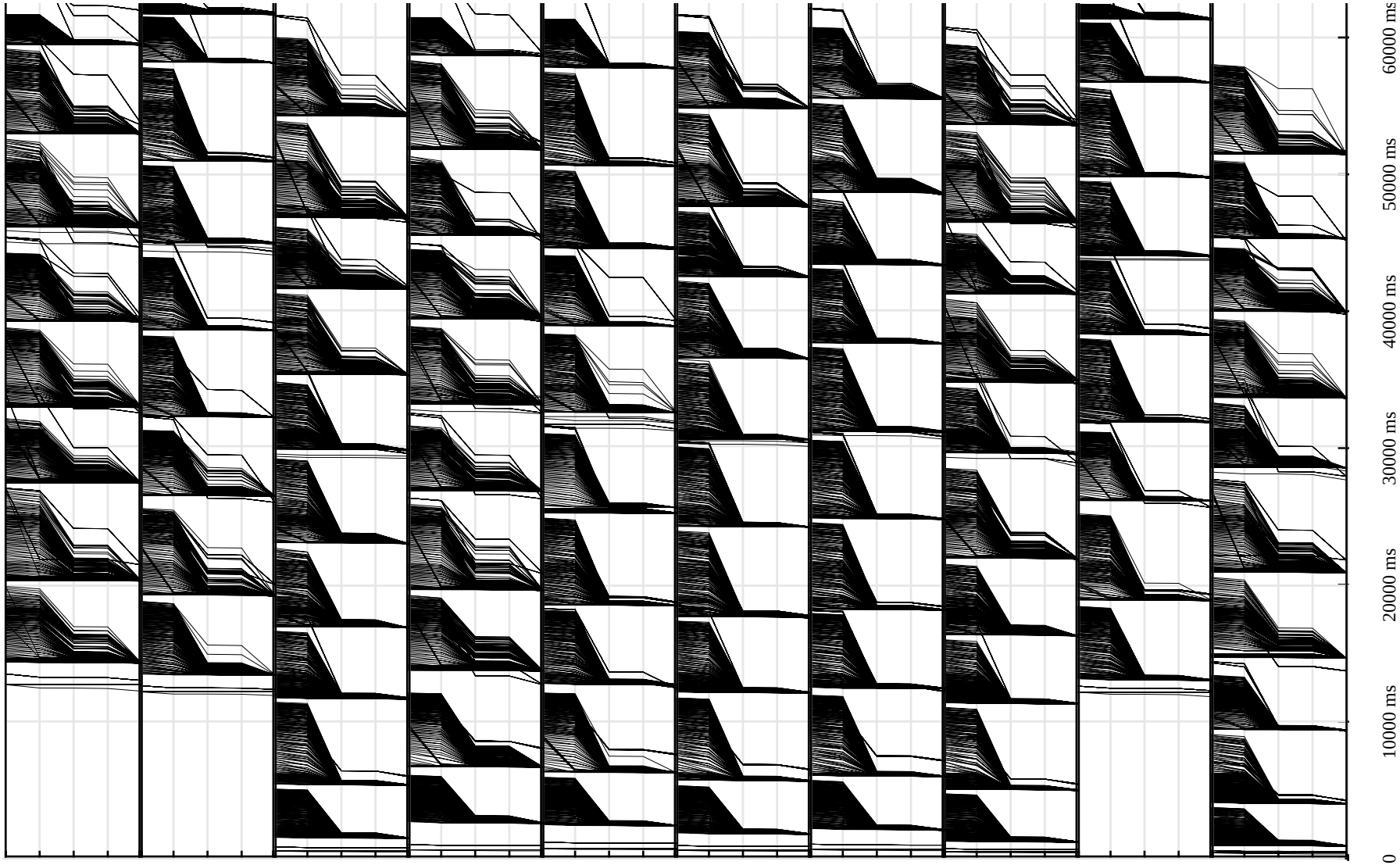
- **data was read in 1 Mbyte units (size of a STAR event) and a total of 1 Gbyte was read**
- **STAF was run on a Sun E-4000 system with an OC-12 (622 Mbit/s) ATM interface**
- **a two server, four disk DPSS configuration was used as the cache**

# Performance

- a data rate of 19 Mbytes/s is achieved for reading data (as expected for two disk servers), and 25 MBytes/s for writing data.

♦ Running 10 instances of the application simultaneously results in the same aggregate throughput with very uniform access, indicating good scalability.

# Performance




**Correct operation of 10 parallel processes reading 10 different data sets from one DPSS (each row is one process, each group is a request for 10 Mbytes of data, total = 1 GBy, *1.5 TBy/day*)**

# Agent Based Management of Widely Distributed Systems

**If comprehensive monitoring is the key to diagnosis (as illustrated below), agent based management may be the key to keeping widely distributed systems running reliably.**

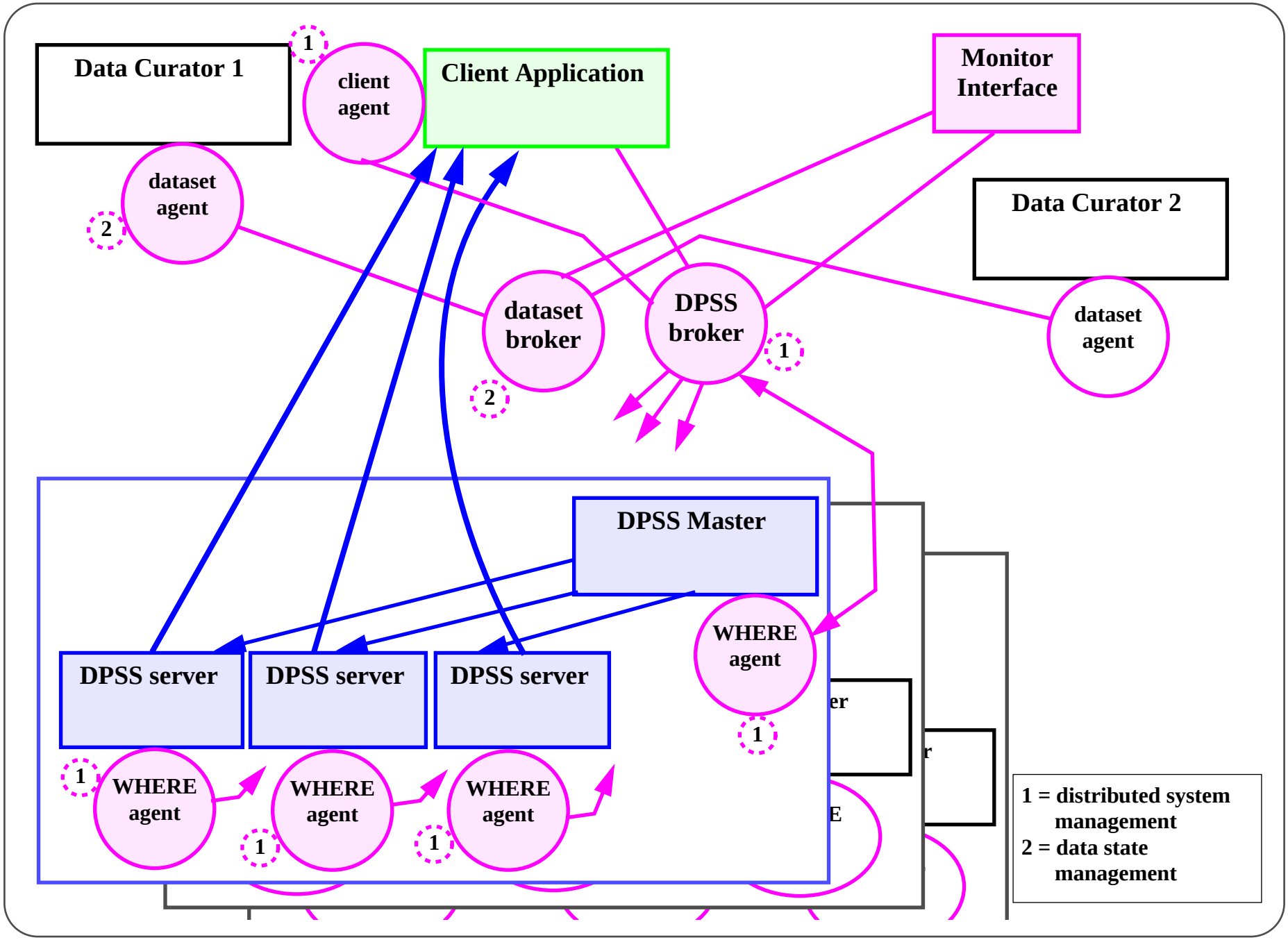
- **"Agents" are**
  - **autonomous**
  - **adaptable**
  - **monitors**
  - **managers**
  - **information aggregates**
  - **KQML based information filters**
  - **implemented in Java**
  - **constantly communicating with peers**

# Agent Based Management

## Initial use of agents

- **Provide structured access to current and historical information regarding the state of the DPSS components**
- **Keep track of all components within the system and restart any component that has crashed, including one of the other agents (addresses fault tolerance)**
- **When new components are added, such as a new disk server, the agents do not have to be reconfigured - an agent is started on the new host, and it will inform all other agents about itself and the new server**
  - ***Brokers and agents may discover interesting new agents via SDR, whence the new agent - and resource that it represents - is added to the configuration. ("Associated" agents communicate with each other using IP multicast.)***

# Agent Based Management



# Agent Based Management

- ***Broker agents*** **manage information from a collection of** ***monitor agents*** **(usually on behalf of a user client)**
  - **Data set state management:**
    - **agents manage dataset metadata (dynamic state, alternate locations tertiary location) at each storage system**
      - **brokers provide an integrated view of the data**
  - **For dynamic configuration and application adaptation**
    - **agents continuously monitor state of all network interfaces and data paths**
      - **brokers analyze this information on behalf of a client to determine which DPSS has the best network connectivity**
    - **agents monitor the load of each DPSS disk server**
      - **broker can analyze to decide which servers to use when data is replicated (addresses high availability)**

# Agent Based Management

- **Brokers can perform actions on behalf of a user**
  - **e.g., if a data set is not currently loaded onto a DPSS (which is typically used as a cache), the broker can cause the dataset to be loaded from tertiary storage**

- **A broker/agent architecture allows the system administrators to separate mechanism from policy**
  - **agent rule-based operation can be used to determine what policies are be enforced while remaining separate from the actual mechanism used to implement these policies**

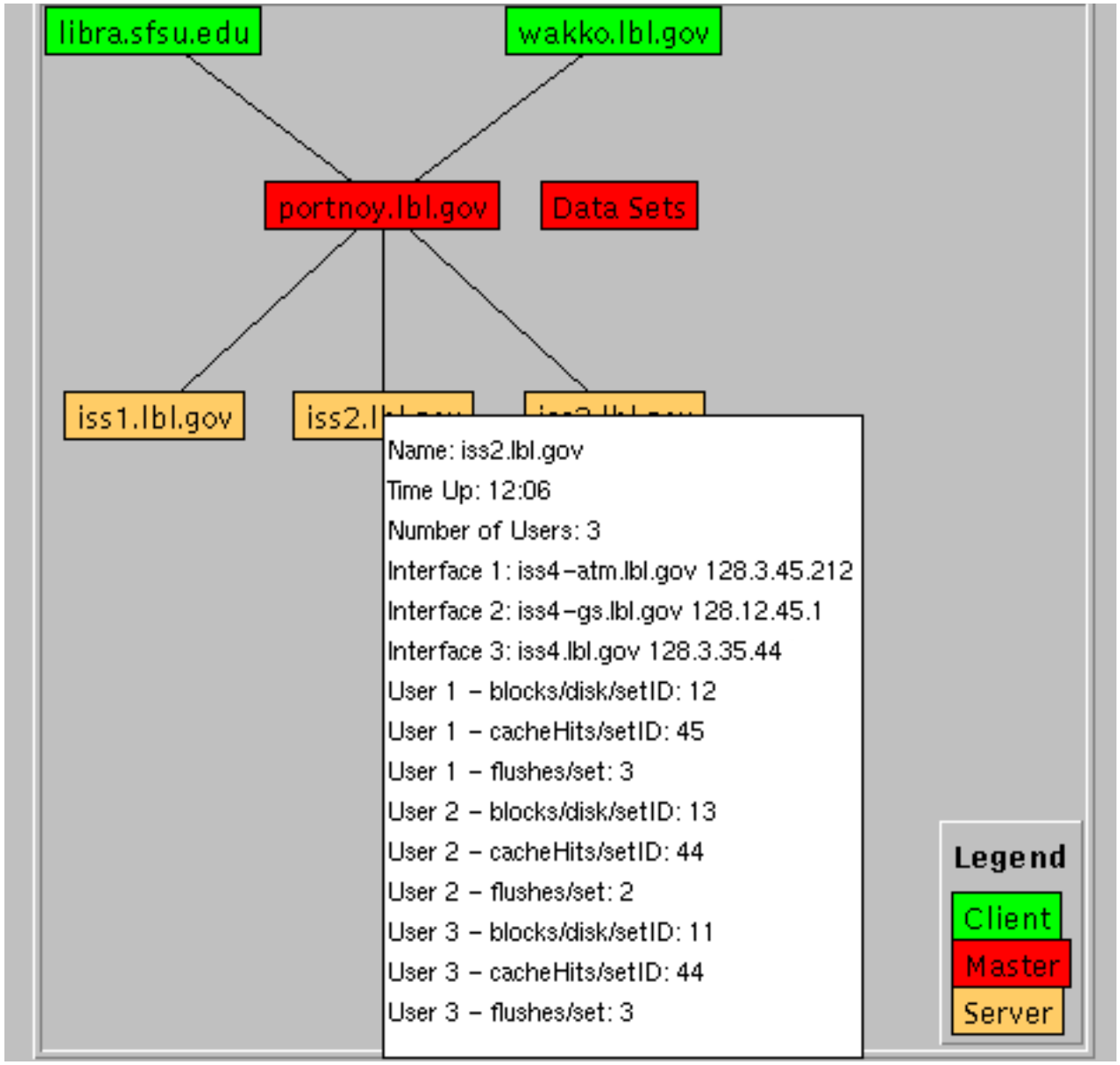
# Agent Based Management

- **New agent methods can be added at any time**
  - **e.g., the brokers have an algorithm for determining which DPSS configuration to use based on a set of parameters that include network bandwidth, latency and disk server load - this algorithm can be modified "on the fly" by loading new methods into the agents**
  - **<u>related agents are part of the same security context</u>, and new code/methods presented to the agents is cryptographically signed for origin verification and integrity**

**<u>First demonstration / experiment</u>**

**In MAGIC, an application will use aggregated information from a broker to present an adaptive and dynamic view of the system: data throughput, server state, and dataset metadata as reported by the agents (versus a directory approach)**

# Agent Based Management




**<u>Prototype automatically generated monitor interface</u> (a Web applet) for brokers aggregating information from DPSS system-state and dataset-state monitoring agents (corresponding to the emphasized elements in the previous figure)**

# Agent Based Management

## Future

- When you observe that something has gone wrong in a widely distributed system, it is generally too late to react - in fact, you frequently can't even tell what is wrong, because
  - it depends on a history of events
  - you can't get at the needed information any more
  - it will take too long to ask and answer all of the required questions

Agents will not only monitor, but keep a state history in order to answer the question "how did we get here?"

- Active analysis of operational patterns (e.g. pattern analysis of data block lifeline traces) will lead to adapting behavior / configuration to avoid or correct problems

# Security

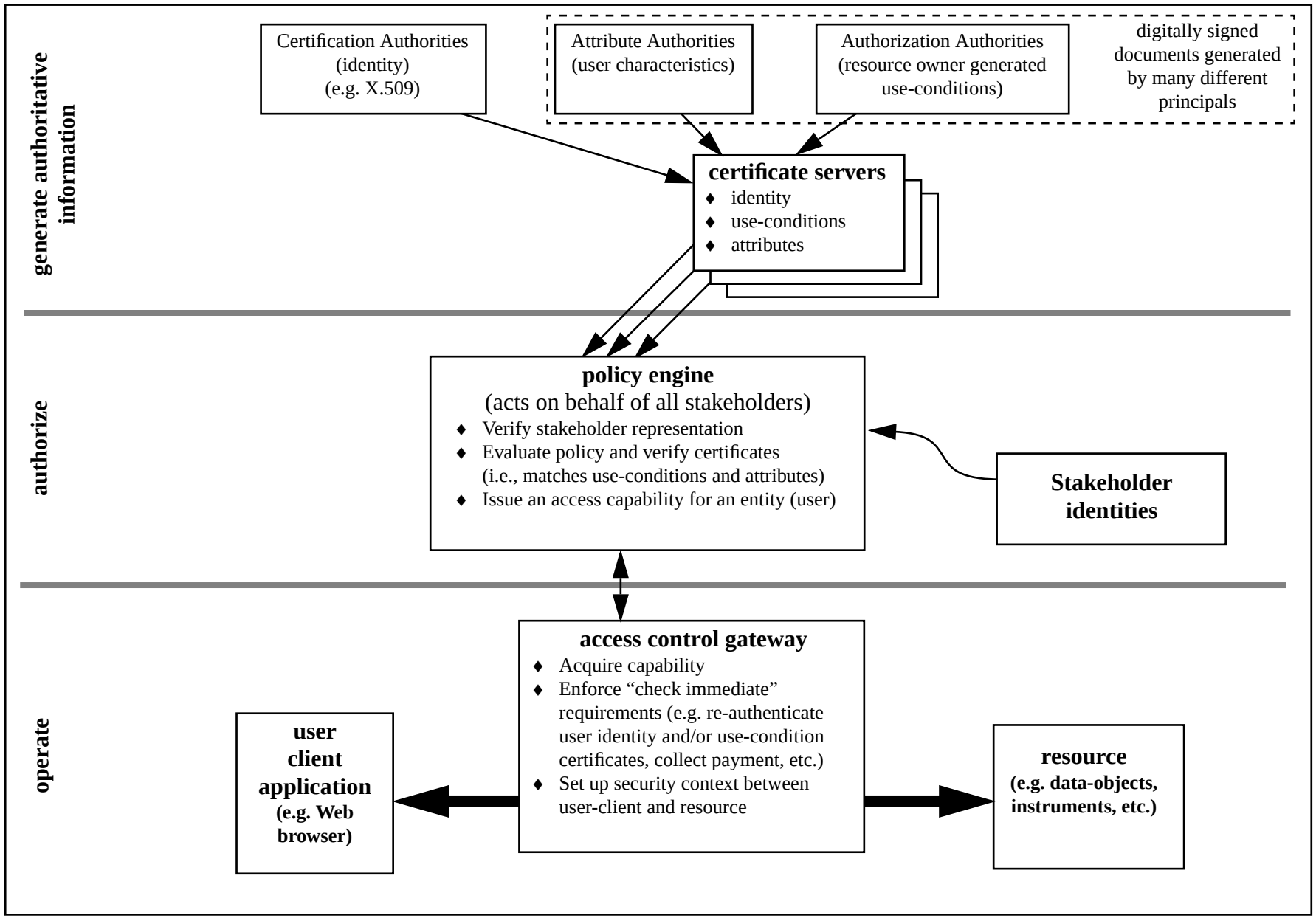
**The overall model for security is that a collection of cryptographically signed are maintained by the responsible party / stakeholder:**

- **authorization (owner / stakeholder use-conditions)**
- **attributes (user characteristics)**
- **identity (of principals - entities and components)**

**represents the access policy in terms of use conditions and corresponding user attributes.**

**Upon an access request, a policy engine collects and evaluates the certificates and returns a yes/no answer. An affirmative answer initiates the context establishment phase of the underlying security mechanism (e.g. SSH or GSS)**

# Security




**Security Model for Widely Distributed Systems**

# An Experiment in High-Speed, Wide Area Distributed Data Handling

**At the data generation end of a physics experiment like those at RHIC, a detector puts out a steady state data stream of 20-40 Mbytes/s. Traditionally, this data is archived and a first level of processing is performed at the experiment site. The resulting second level data is also archived and requested later for the analysis described above. This results in the data being archived at the experiment site in "medium" sized tertiary storage systems.**

**It is our contention that in the time frame of the next generation of physics experiments (2000-2005 AD) that wide area networks will be easily capable of distributing the instrument output data stream anywhere in the US (and probably to Europe).**

**There are two advantages to this scenario. First, the first level processing (which is easily parallelized) can be done using resources at the collaborators sites (each experiment typically involves 5-10 major institutions). Second, large tertiary storage systems exhibit substantial economies of scale, and so using a large tertiary storage system at, say, a supercomputer center, should result in more economical storage, better access (because of much larger near-line systems - e.g. lots of tape robots) and better media management, especially in the long term, than can be obtained in local systems.**

**To this end, we propose that the data handling architecture developed for the real-time**

# Wide Area Distributed Data Handling

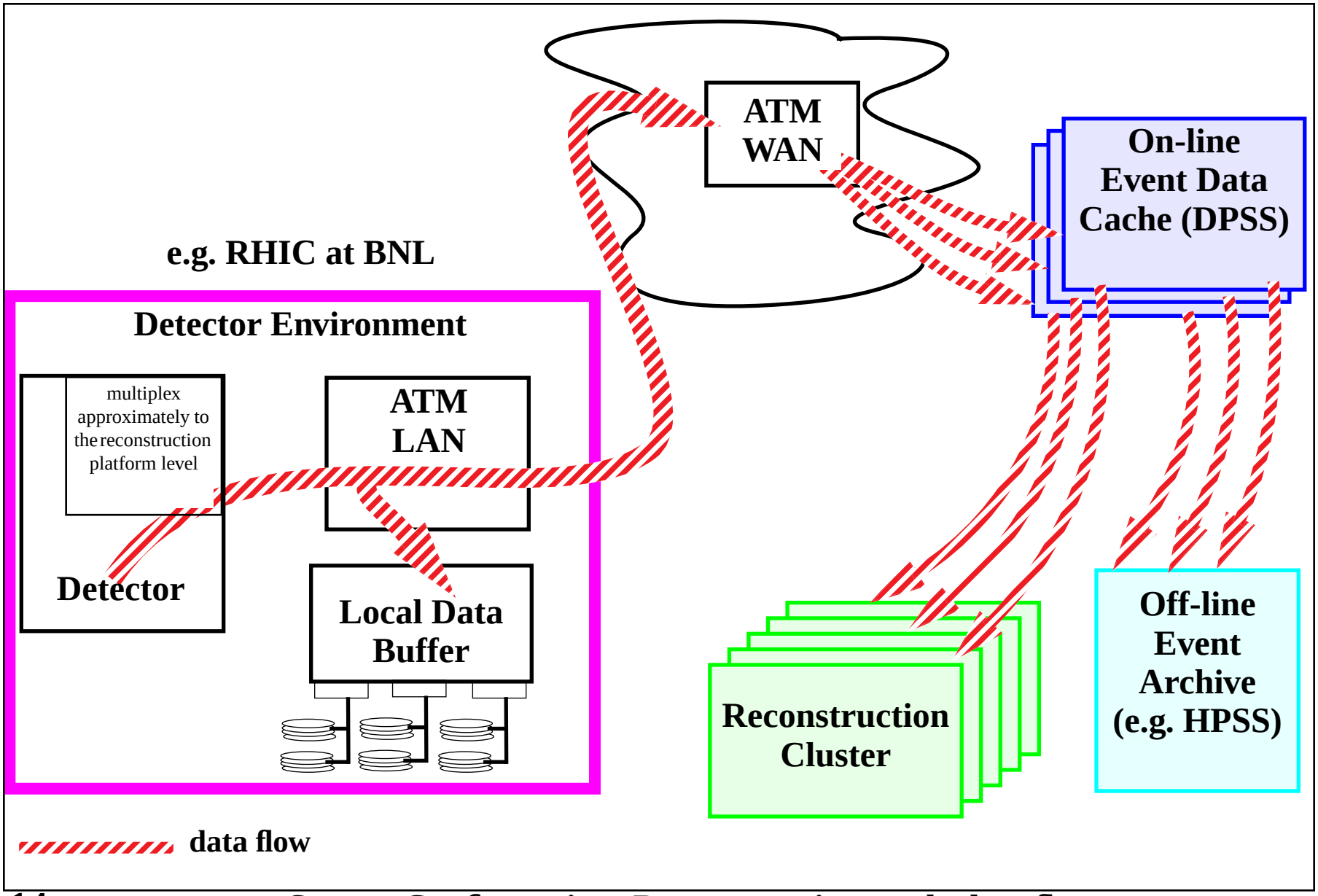
**digital cataloguing system noted above (Figure 2) can be used for this purpose.**

**For the first level data from the instrument system, the data flows from the instrument, through the network, to a receiving cache. (See figure 14) The DPSS is used for this cache, the servers of which may be located at one, or (as in the MAGIC testbed) distributed across many sites. The first level of processing can be done directly out of the cache. The first level data is also moved from cache to tertiary storage, and the results of this processing can be used to optimize data placement on tertiary storage.**

**This scenario is being tested in the following experiment. A DPSS and a computing cluster are located at Lawrence Berkeley National Laboratory. The NTON network testbed that connects Berkeley and LLNL can be configured for a 2000 km, OC-12, path. A high-speed workstation that has a collection of STAR events stored on its disks is located at LLNL and connected to NTON. This workstation will emit events at the same rate as the STAR detector, and this data will be cached on the DPSS at Berkeley. The computing cluster will process data out of the cache (doing "reconstruction") and those results will be written back to the cache. A storage manager will migrate data to tertiary storage (or a "null" system that has the same throughput characteristics, as there is little point in actually storing this synthetic data).**

**Except that the computing cluster will not have sufficient compute capacity to do all of the required processing at the operating data rates, this scenario - once it works as expected - should demonstrate the feasibility of wide area processing of this type of**

# Wide Area Distributed Data Handling

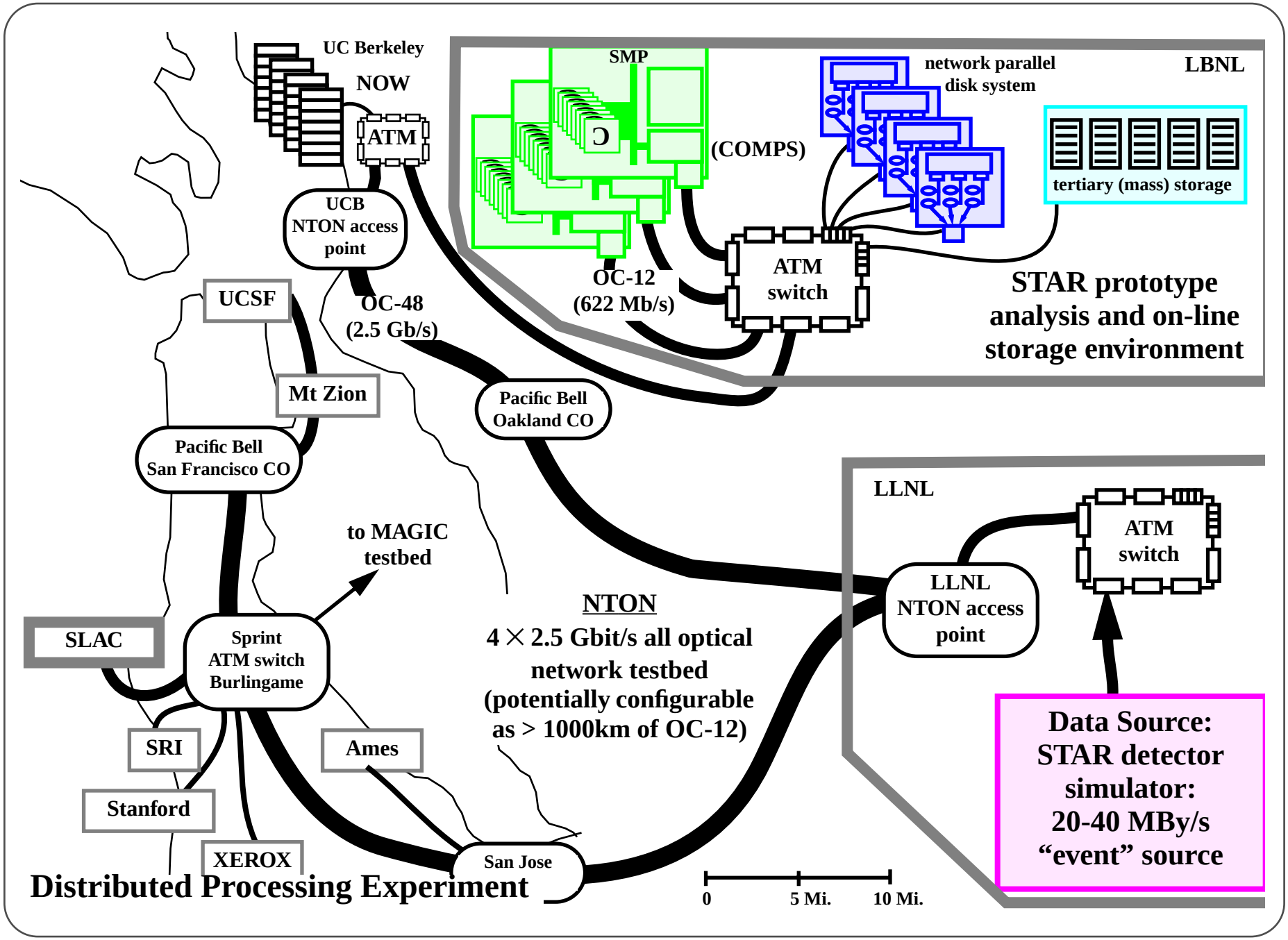



**14.** System Configuration: Reconstruction mode data flow

# Wide Area Distributed Data Handling

**real-time data. The experiment is illustrated in figure 15.**

# Distributed Processing Experiment

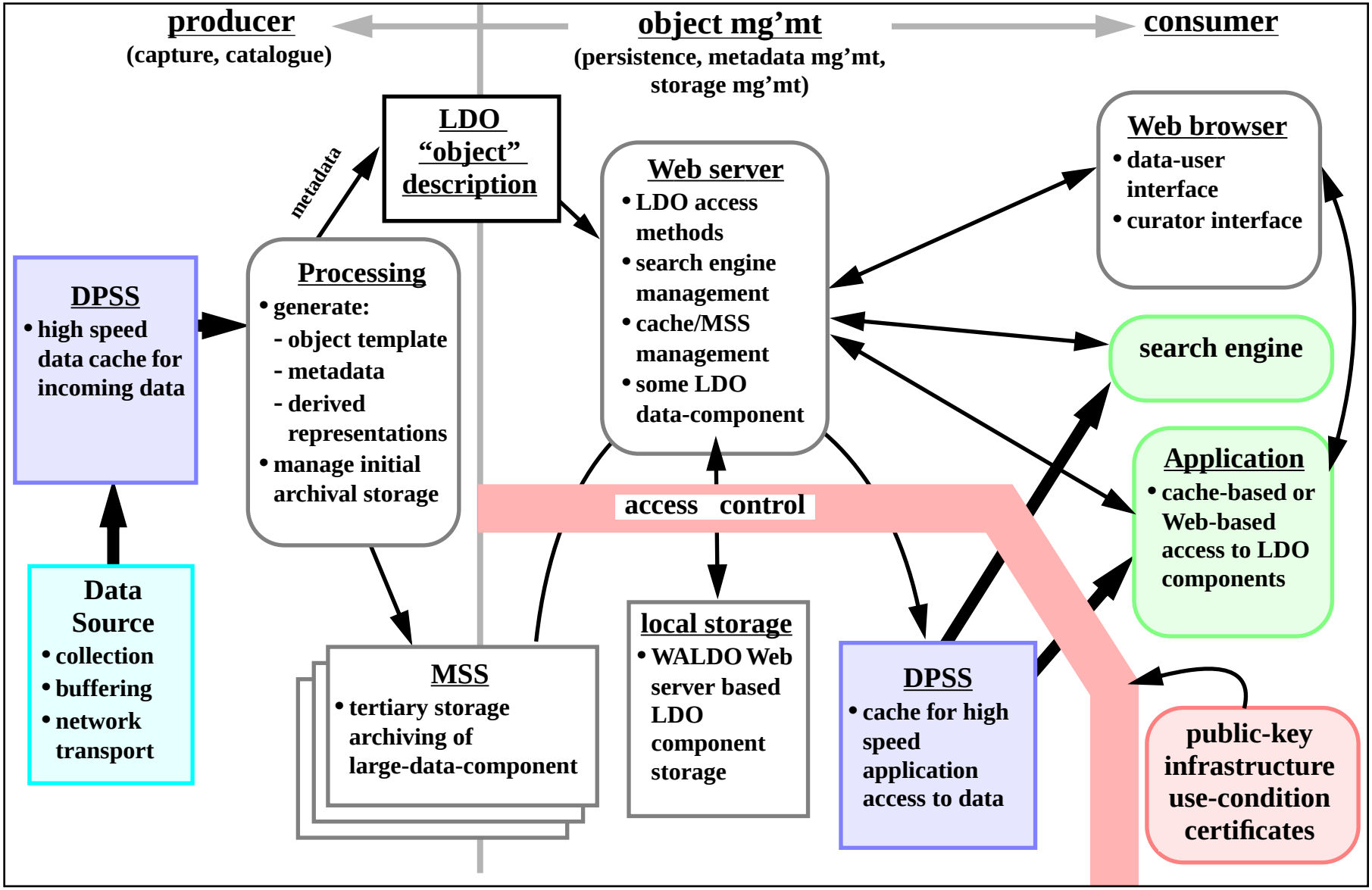


# The Wide Area, Large Data Object System

- **WALDO was the first system in which we used the DPSS for high-speed data collection and on-line distributed processing of that data.**

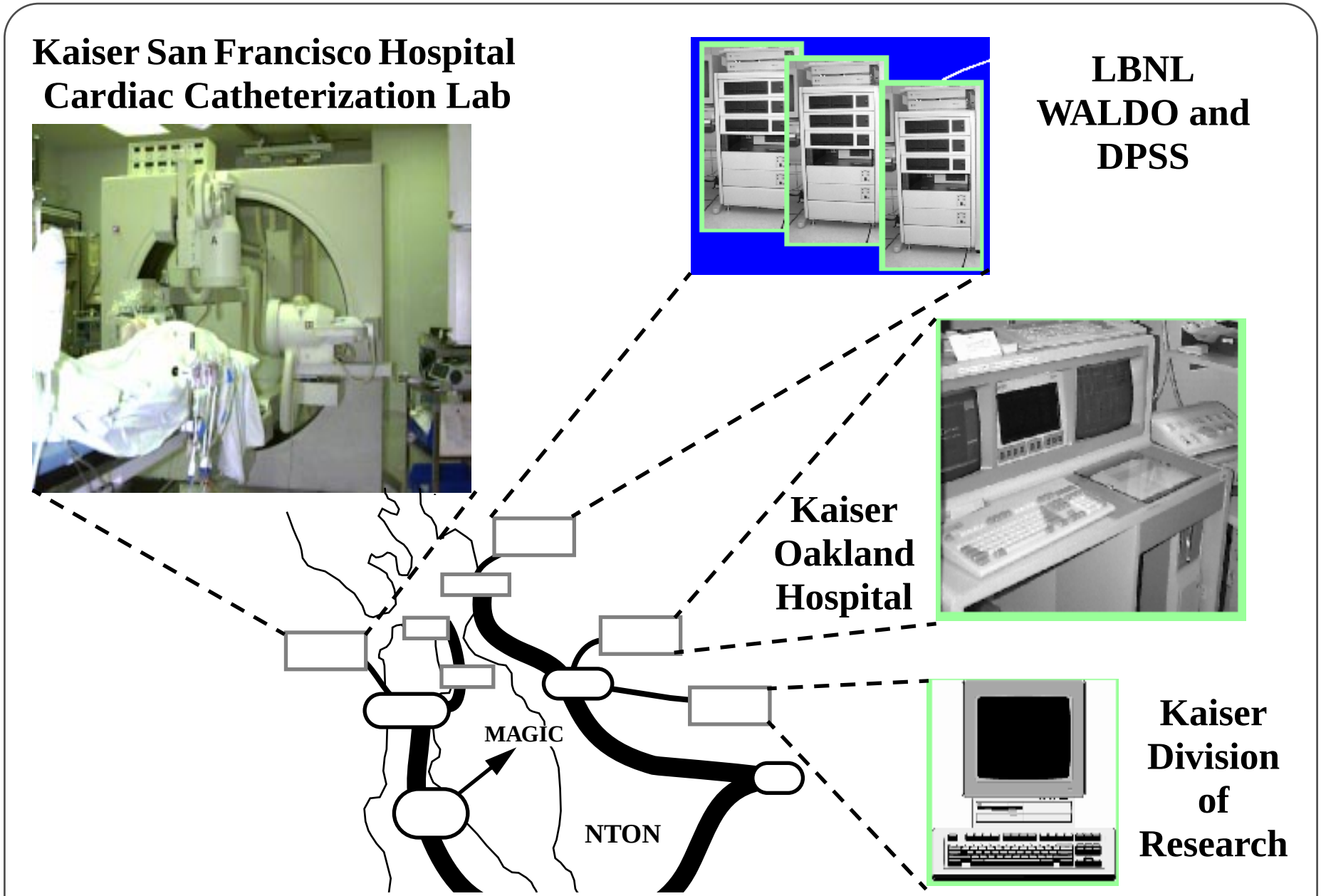
  **This general model has been used in several data-intensive computing applications. For example, a real-time digital library system (see figure 2 and [DIGLIB]) collects data from a remote medical imaging system, and automatically processes, catalogues, and archives each data unit together with the derived data and metadata, with the result being a Web-based object representing each dataset. This automatic system operates 10 hours/day, 5-6 days/week with data rates of about 30 Mbits/sec during the data collection phase (about 20 minutes/hour).**

# WALDO



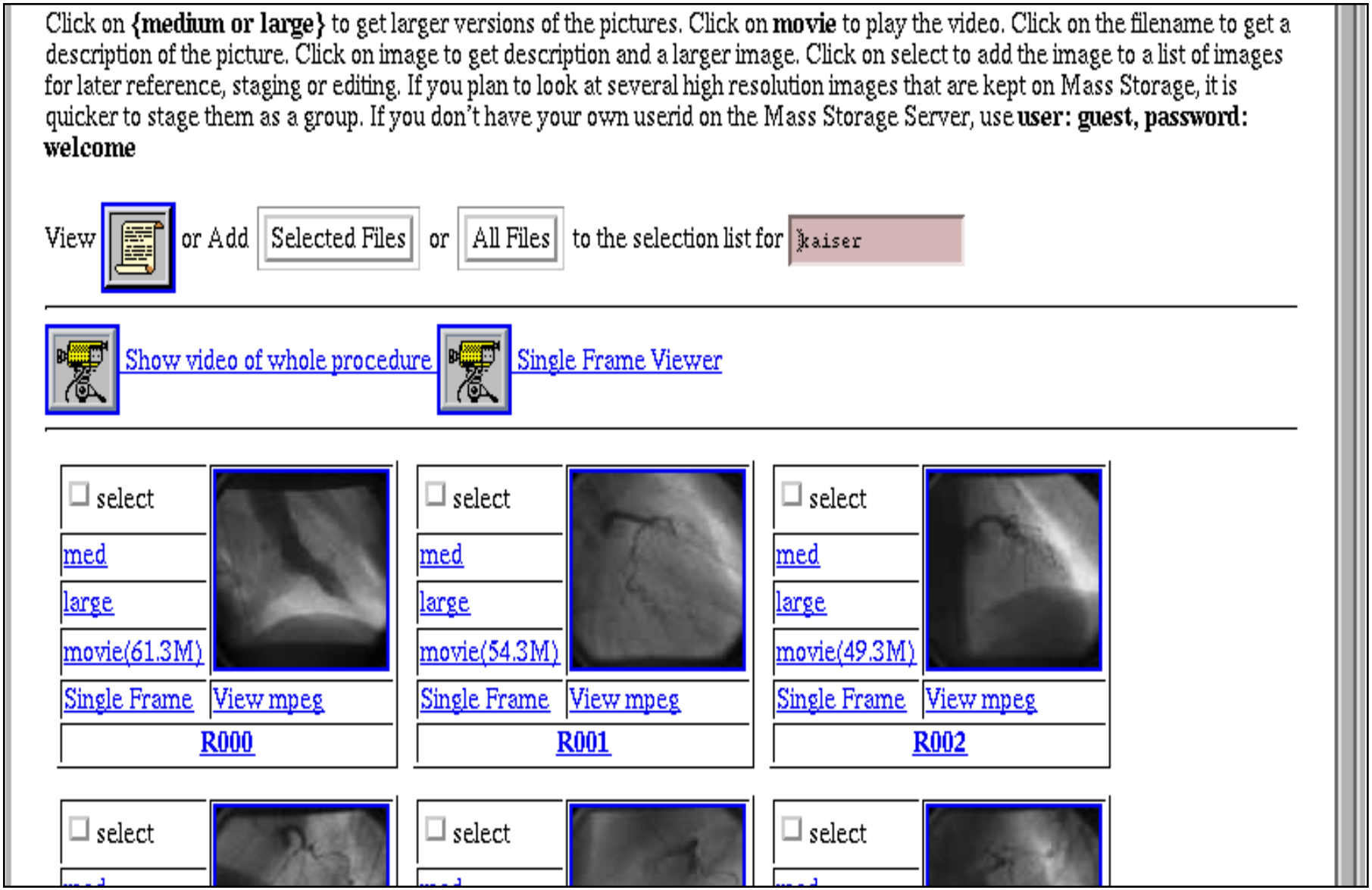

2. WALDO Data Flow for Automatic Digital Library Generation

# WALDO




3. Kaiser / LBNL Health Care Imaging System and NTON Testbed

# WALDO




**4. A representation of the six objects (about 0.75 GBy total) resulting from a single cycle of operation of a remote, on-line cardio-angiography system**

# Conclusions

**The experiments described here are work-in-progress. The use of the DPSS as cache has demonstrated the required performance, but a complete demonstration of scalability requires running hundreds of analysis processes. The wide area, high-data rate experiment configuration is nearly complete, and results are expected in the near future. We expect that this experiment will be successful, because several precursors have been carried out in the MAGIC testbed. However, experience has also shown that every significant increase in throughput and/or scale raises a new set of issues.**

# References and Notes


**COMPS** "Clusters of Multi-Processor Systems in the Scientific Environment". See http://www-itg.lbl.gov/~johnston/COMPS/

**[DIGLIB]** "Real-Time Generation and Cataloguing of Large Data-Objects in Widely Distributed Environments", W.Johnston, Jin G., C. Larsen, J. Lee, G. Hoo, M. Thompson, B. Tierney, J. Terdiman. To be published in International Journal of Digital Libraries - Special Issue on "Digital Libraries in Medicine".

**DOE2000** See http://www-itg.lbl.gov/DCEE

**DPSS** "The Distributed-Parallel Storage System (DPSS)". See http://www-itg.lbl.gov/DPSS.

**Lau94** "TerraVision: a Terrain Visualization System". S. Lau, Y. Leclerc, Technical Note 540, SRI International, Menlo Park, CA, Mar. 1994. Also see: http://www.ai.sri.com/~magic/terravision.html .

**MAGIC** "The MAGIC Gigabit Network" (http://www.magic.net/)

**NTONC** "National Transparent Optical Network Consortium". See http://www.ntonc.org . (NTONC is a program of collaborative research, deployment and demonstration of an all-optical open testbed communications network.)

**STAR1** "Relativistic Nuclear Collisions Program", H.G. Ritter. http://www-library.lbl.gov/docs/LBNL/397/64/Overviews/RNC.html

**STAR2** "High Speed Distributed Data Handling for HENP", W. Greiman, W. E. Johnston, C. McParland, D. Olson, B. Tierney, C. Tull. http://www-rnc.lbl.gov/computing/ldrd_fy97/henpdata.htm

**Thomp97** "Distributed Health Care Imaging Information Systems". M. Thompson, W. Johnston, G. Jin, J. Lee, B. Tierney, Lawrence Berkeley National Laboratory, Berkeley CA, and Terdiman, J. F., Kaiser Permanente, Division of Research, Oakland CA. SPIE International Symposium on Medical Imaging, 1997. Newport Beach, California. (Also available at http://www-itg.lbl.gov/Kaiser.IMG/homepage.html)

**Tull97** "The STAR Analysis Framework Component Software in a Real-World Physics Experiment". C.Tull, W.Greiman, D.Olson, D.Prindle, H.Ward, International Conference on Computing in High Energy Physics, Berlin, Germany, April, 1997.

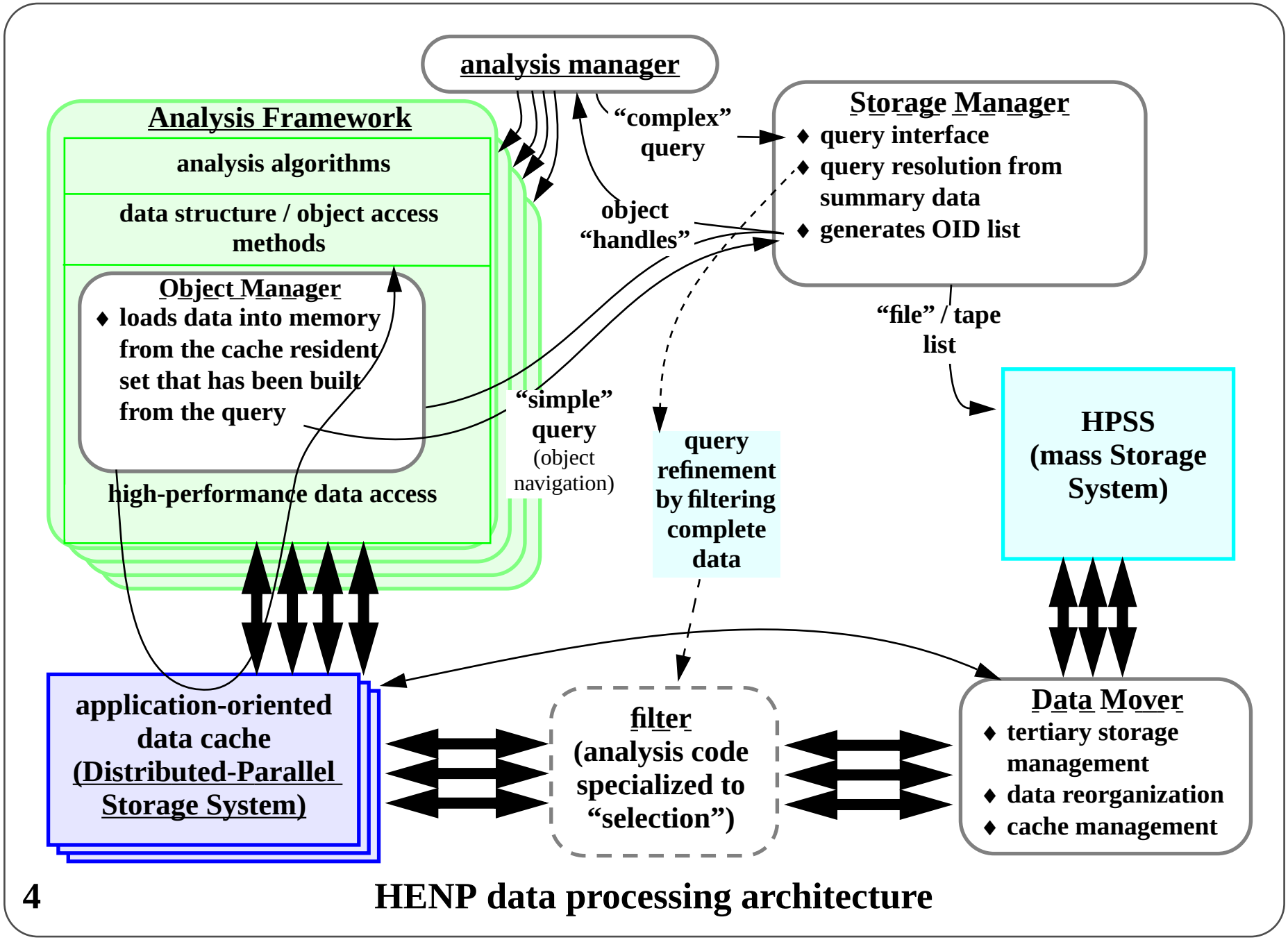
**[Tierney]** "Performance Analysis in High-Speed Wide Area ATM Networks: Top-to-bottom end-to-end Monitoring", B. Tierney, W. Johnston, J. Lee, G. Hoo. IEEE Networking, May 1996.


- **http://www-itg.lbl.gov/~johnston/CERNSchool**
- **http://www-itg.lbl.gov/STAR**

## Application Architecture (figure 4):

- An analysis management and user interface system generates queries that produce a list of objects of interest. (1), (2)
- This list of objects, then, has to be retrieved from tertiary storage (3), and loaded into the cache for processing (6). The cache loading process involves parallel transfers from the tertiary storage system to the cache.
- A data mover (4) reads data from the tertiary storage system and reorganizes it for placement on the cache.
- An “in-line” filter that uses a specialized version of the analysis code may refine the query results. (5)

- When a dataset (or part of a dataset) has been loaded into the cache, the object manager is notified, and it in turn notifies the analysis code.
- Multiple instances of the analysis code (operating under the control of a analysis manager) running simultaneously on many different systems then read data from the cache into memory, and processing commences. (7)
- In the prototype architecture the analysis systems may be widely distributed, and they all consume data from the cache, and return results to the cache.
- Two approaches to managing data within the analysis code are illustrated: STAF, which manipulates self-describing tables, and an object-oriented database, which manages C++ objects.




4 **HENP data processing architecture**