

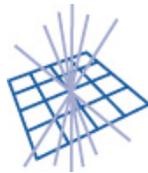
# PhEDEx

## Physics Experiment Data Export

Tim Barrass (Bristol University, UK)  
On behalf the PhEDEx development team

*Technical meeting at BNL*

*4 April 2005*



**GridPP**  
UK Computing for Particle Physics



University of  
**BRISTOL**

# Talk structure

- What is PhEDEx?
  - An overview of current deployment and usage
- PhEDEx in context
  - What it needs to do for CMS
  - Data management for other HEP experiments
  - EGEE and other grid projects
- PhEDEx in detail
  - The design of PhEDEx
  - Management of dataset-scale transfers
  - Multi-hop file routing
  - Reliable point-to-point transfers
- Future work on PhEDEx
  - Contractual file routing, peer-to-peer data location, semi-autonomy, policy and priority, new technologies ...



What is PhEDEX?

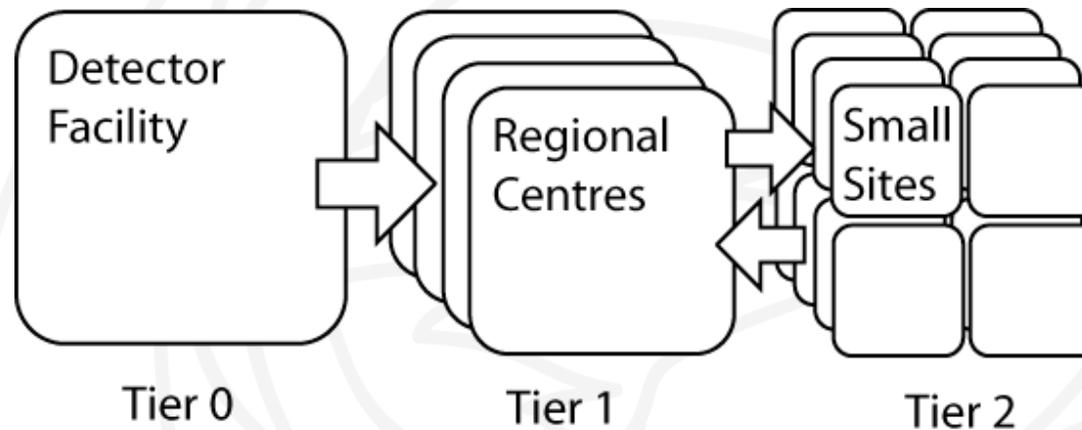
## What is PhEDEx?

# Introduction

- PhEDEx is the data file replica management system used by CMS
- Simple twofold goal
  - Manage the prioritized transfer of files from multiple sources to multiple sinks
  - Provide information on cost- latency and rate- of any given transfer to enable scheduling
- Enables CMS to manage the distribution of data at dataset level rather than at file level
- Bridges the gap between “traditional” and “Grid” data distribution models
  - Traditional  $\Rightarrow$  large-scale transfers between large sites, often managed by hand
  - Grid  $\Rightarrow$  replication of data in response to user demand
- Manages multi-hop transfers through not-completely-connected distribution networks
  - A core, stable infrastructure handling large-scale continuous transfers
  - A dynamic, Grid-like infrastructure associating with the core

# What is PhEDEx?

## CMS data flows

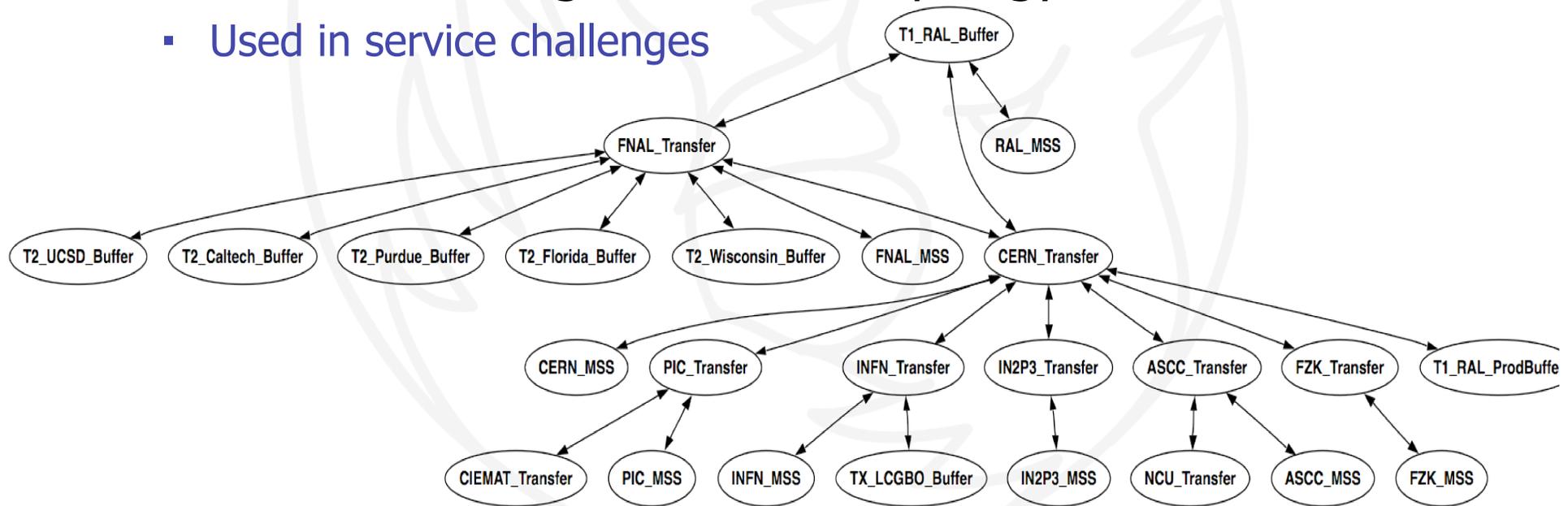


- Detector data flows to Tier 1 sites
  - Stored safely to tape
  - Undergoes large-scale processing and analysis
- Processed data flows to Tier 2 sites
  - Undergoes small-scale analysis
- Simulation and analysis results flow from Tier 2 sites
  - Cached at Tier 1s
- Core infrastructure is a stable set of Tier 0, Tier 1 and Tier 2 sites
- Dynamic infrastructure typically Tier 2 and smaller sites that are transient
  - Each associating with a larger site

# What is PhEDEx?

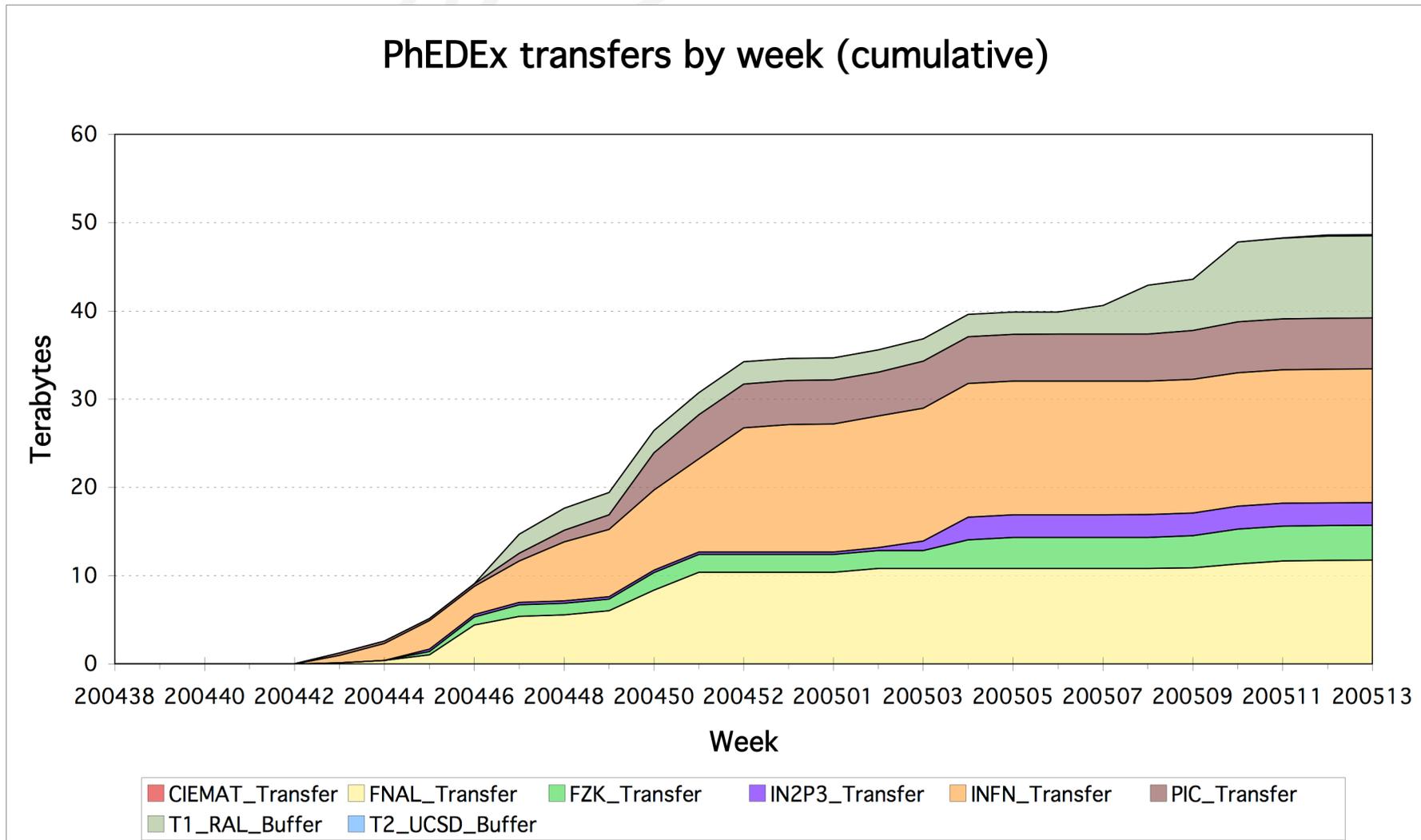
## Current deployment

- Production PhEDEx deployed at 7 large sites
  - FNAL, CERN, INFN-CNAF, PIC, RAL, FZK, IN2P3
  - Also at a number of smaller (T2) sites
    - Florida, UCSD (US); Imperial (UK); INFN-Bologna (IT) ...
- Also other sites registered with topology
  - Used in service challenges



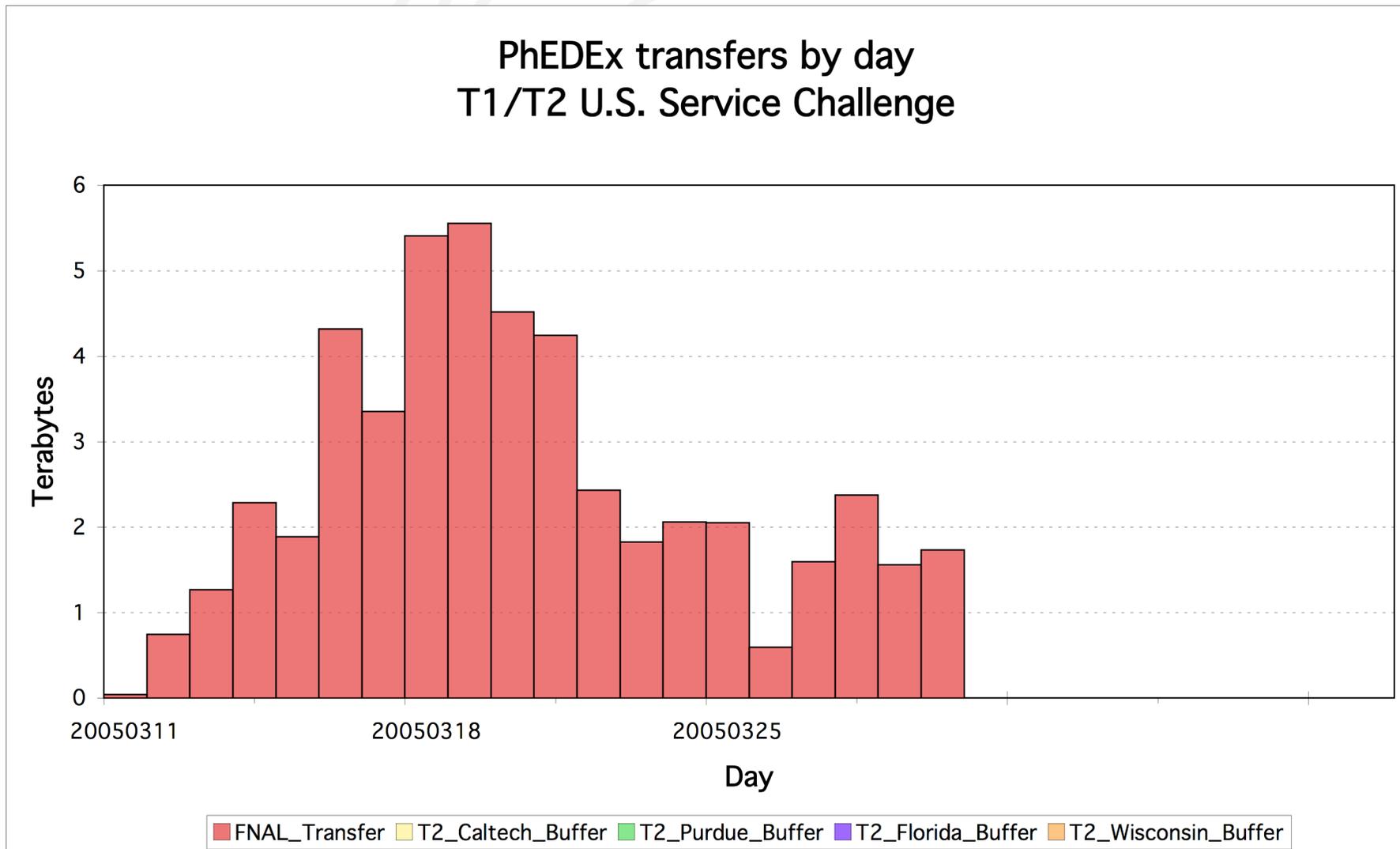
# What is PhEDEx?

## Current usage for production



What is PhEDEx?

# Current usage in LCG Service challenge



What is PhEDEx?

## Summary and status

- PhEDEx manages large-scale transfers for CMS
  - Large-scale  $\Rightarrow$   $O(1000+)$  files per dataset
  - In development and production for about a year
  - Currently version 2.1
- In production operation now
  - $\sim 70$  TB known to PhEDEx,  $\sim 150$  TB total replicated
  - Not all links are bidirectional yet
    - All Tier-1s operational for inbound transfers, CERN and FNAL have demonstrated data export, a few others starting testing
  - Some Tier-2+ can do inbound transfers (CIEMAT, INFN Bari, UCSD), several others installing and/or testing (U.S., Italy, Finland, UK)
  - Reaching nearly 20TB a month
  - Main issues are with underlying fabric- more later
- Also being used as part of the current LCG robust data transfer service challenge
  - Reaching up to 5TB a day



PhEDEX in context

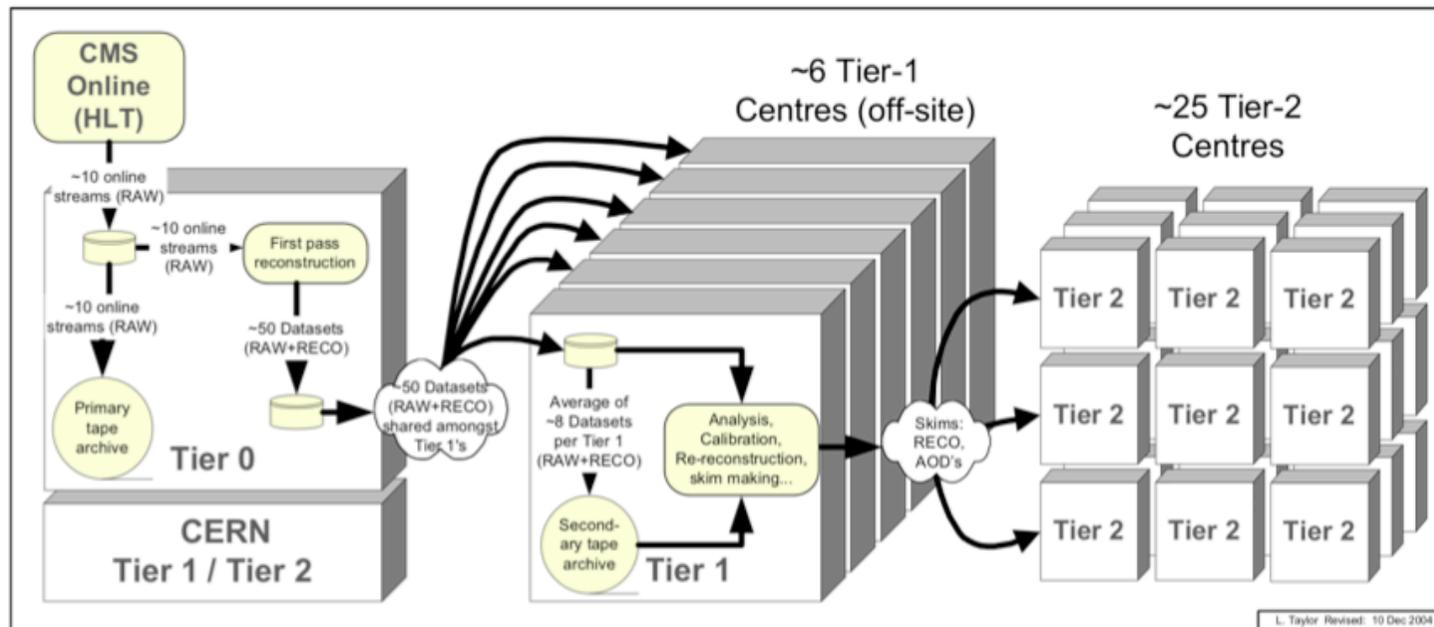
## PhEDEx in context

# Introduction

- HEP experiments have found many ways of solving the problem of distributing their data
- Seems that no other system meets CMS' requirements
  - LHC implies a significant ramp up in rate of data movement
    - 10 PB a year for CMS alone
  - CMS also has many components already in existence
    - Which are replicated/preceded by other systems in ways that make it non-trivial to integrate?
- That said we actively seek contact with other groups managing data distribution
  - Lots of experience out there
- Put PhEDEx in some context
  - CMS requirements
  - Other systems
  - EGEE ...

# PhEEx in context: CMS

## CMS data flows revisited



- Rate from online to offline computing  $\sim 225$  MB/s
  - 3 day buffer needs to be 56 TB
- Each Tier 1 site required to handle
  - $\sim 6$  Gb/s sustained incoming
  - $\sim 4$  Gb/s sustained outgoing
  - *Some* efficiency factors here to enable clearing of filling buffers after downtime, etc

PhEEx in context: CMS

# CMS requirements

- Push from detector facility
  - Irreplaceable raw detector data to tape at Tier 1
- Pull to requesting sites
  - Raw and processed data from Tier 1
  - Simulated data from producing sites
- Not a low level description of operation
  - Many arguments for always pulling data
  - Instead- a description of which body within CMS is initiating the transfer
    - Collaboration as a whole
    - Physics analysis group; simulation production managers ...
    - Single analyst downloading to laptop
- Resolution of competing demands an issue
  - Local and global priorities need to be resolved by policy
- Distribution topology is not fully-connected
  - Rather, it's more a hybrid of tree-mesh-star ...

## PhEEx in context: CMS

# CMS requirements

- Tiered data flow partly structured to manage load on resources
  - Prevent everyone connecting to detector facility
- Tier 1 sites have a sophisticated role
  - Permanent safe storage of a copy of raw data subset
    - Accommodating unmanaged downtime of peers to ensure data security
  - Serving of raw and reprocessed data to associated Tier 2 sites
    - May have been reprocessed at a peer site
  - Caching of data produced at associated Tier 2 but destined for other sites
    - Permanently or temporarily
- In detail these involve managed multi-hop transfers
  - Tape > disk > disk > tape/disk > disk
  - Need to manage the cleaning of buffers at detector facility- only delete when safe at N Tier 1 sites
  - Need to determine when files produced at Tier 2 have reached all destinations of interest, or cached at Tier 1
- Need to maintain a view of replica state in detail
  - e.g. Has this *actually been stored on tape and checksummed*

## Other systems

- SAM(Grid) for CDF, D0
  - Strongly couples many aspects of experimental operation
    - Dataset bookkeeping and auditing
    - Transfers
    - Workload management
  - Large scale data movements handled
    - N main sites
  - Moves data in response to user demand
- EDG for LHC experiments and others
  - Much research into optimized replica management in response to demand
  - No production-quality automated data management
    - Still only point-to-point, download-your-own
- CondorG (+Stork)
  - Again, coupled workload and data management
  - No automated data management- no sense of continuous background data flow in its own right
- ATLAS- Don Quixote, and the reliable file transfer service?
  - Parallel development with slightly different emphasis in detail?

# PhEDEx in context: wider environment

## EGEE gLite project

### Replica management

Simple allocation of files to destinations based on subscriptions.  
Determination of closest/ best replica for transfer.  
More grid-like "global" replica management based on demand

### Dataset/ chunk level transfer

Monitor transfers at chunk level, notify sites on progress.  
Activate and deactivate chunks.  
Alter routing dynamically to avoid problems.  
Handle automatic harvesting of files and bulk transfer requests

### Reliable routed, or multi-hop, transfer

Efficient handover of responsibility from node to node in a transfer chain.  
Manage clustering of tape stages and migrations.

### Reliable point-to-point, or single hop, transfer

Failure recovery and retry of transfers.

### Unreliable point to point transfers and technologies

srmcp, globus-url-copy, lcg-rep, dccp  
srm, gsiftp, dCache

CMS  
specific  
management  
layers

EGEE gLite  
File Transfer  
Service?



PhEDEx in detail

PhEDEx in detail

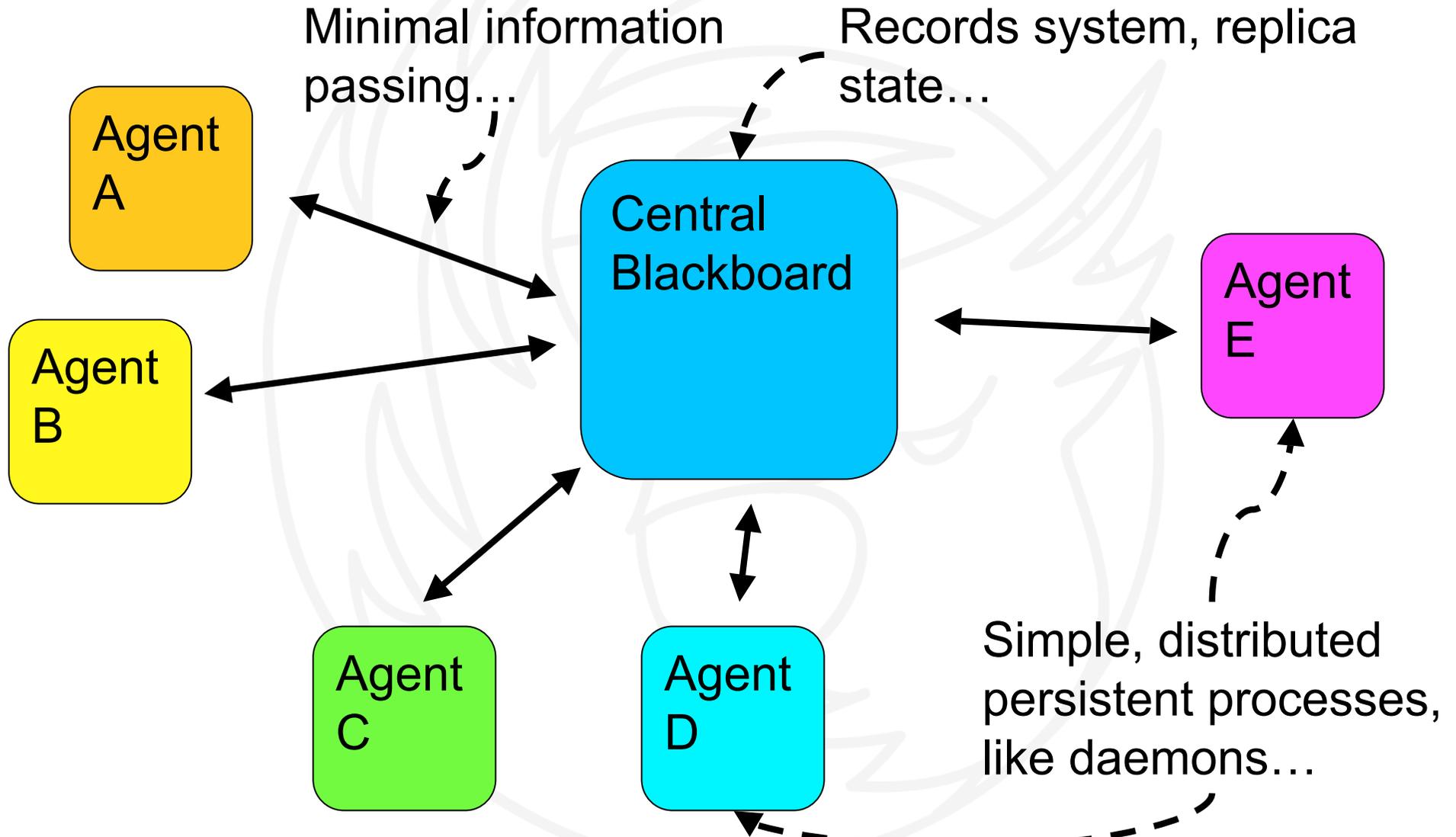
# Introduction

- Design
  - The overall structure of PhEDEx
- Dataset-level management
  - And general deployment notes
- Routed multi-hop transfers
  - Maintaining network topology, choosing best replicas, the need to resolve global and local policy
- Reliable point-to-point transfers
  - Handshaking between components to ensure safe replications

# Design

- Keep complex functionality in discrete units
  - Handover between functional units minimal
- All components should be lightweight and disposable
  - Components defined only by their functionality- and their interaction with other components
- Layered abstractions make system robust
- Keep local information local where possible
  - Enable site administrators to maintain local infrastructure
  - Robust in face of most local changes
    - Deletion and accidental loss require attention
- Leverage hierarchy of data groups to improve performance
  - Easier to manage a dataset than  $O(10000)$  files
- Draws inspiration from agent systems, “autonomic” and peer-to-peer computing

# Agents and blackboards



## Agent message passing

- No messages passed directly between agent processes
  - Instead messages posted on blackboard, read asynchronously by other agents
  - Most agents don't know that anyone else exists
- Transaction-safe passing of local state information down chains of local agents
  - Packets of information persisted as "drops"
  - Placed in "inbox", processed, placed in "outbox" and transferred to next in chain
  - Examples in injection of files into PhEDEx; on-arrival processing and monitoring
    - Typically the points at which another domain links to PhEDEx

# Layers

## **Request management- dataset level transfers**

Scalable management and monitoring of transfer requests.  
Automated allocation of files to destinations to fulfill requests.  
Dynamic routing alterations to avoid problems.  
Automatic harvesting of files; bulk transfer requests for existing data.

## **Reliable routed, or multi-hop, transfer**

Efficient handover of responsibility from node to node in a transfer chain.  
Manage clustering of tape stages and migrations.  
Determination of closest/ best replica for transfer.

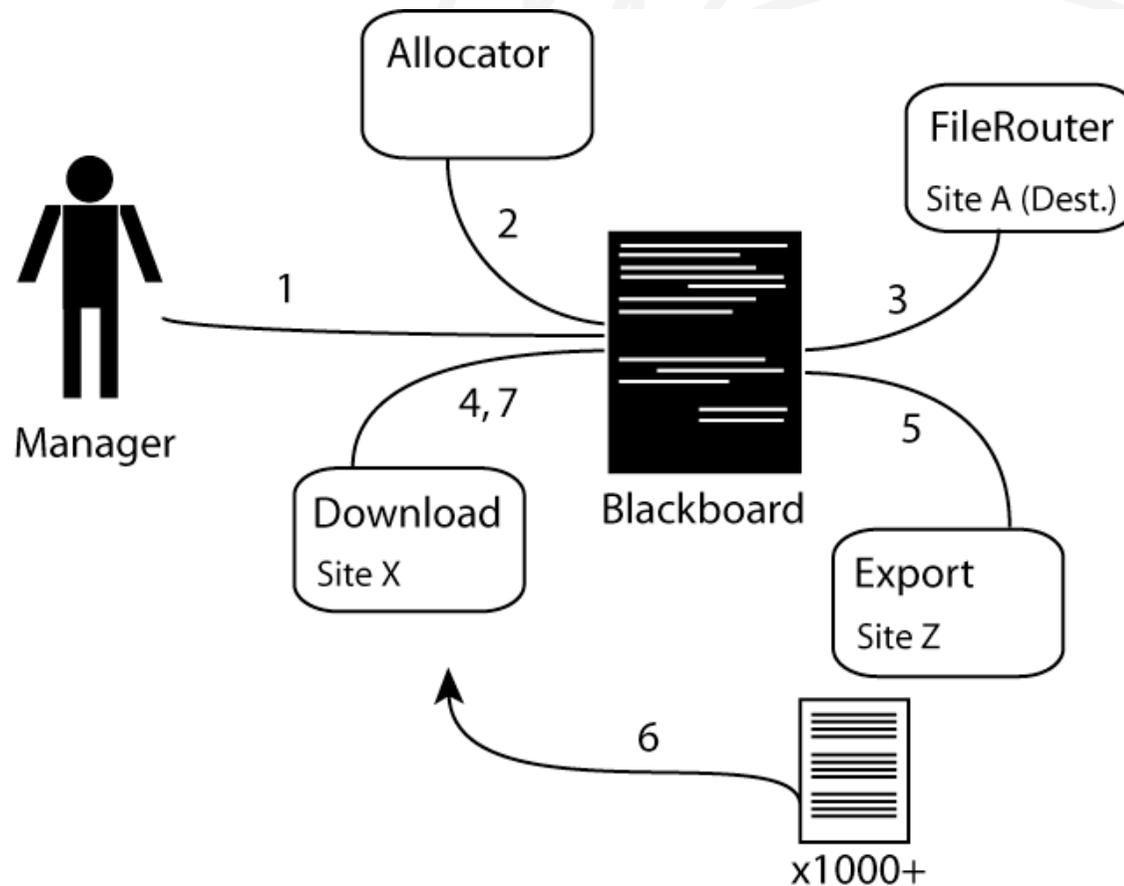
## **Reliable point-to-point, or single hop, transfer**

Failure recovery and retry of transfers.

## **Unreliable point to point transfers and technologies**

srmcp, globus-url-copy, lcg-rep, dccp  
srm, gsiftp, dCache

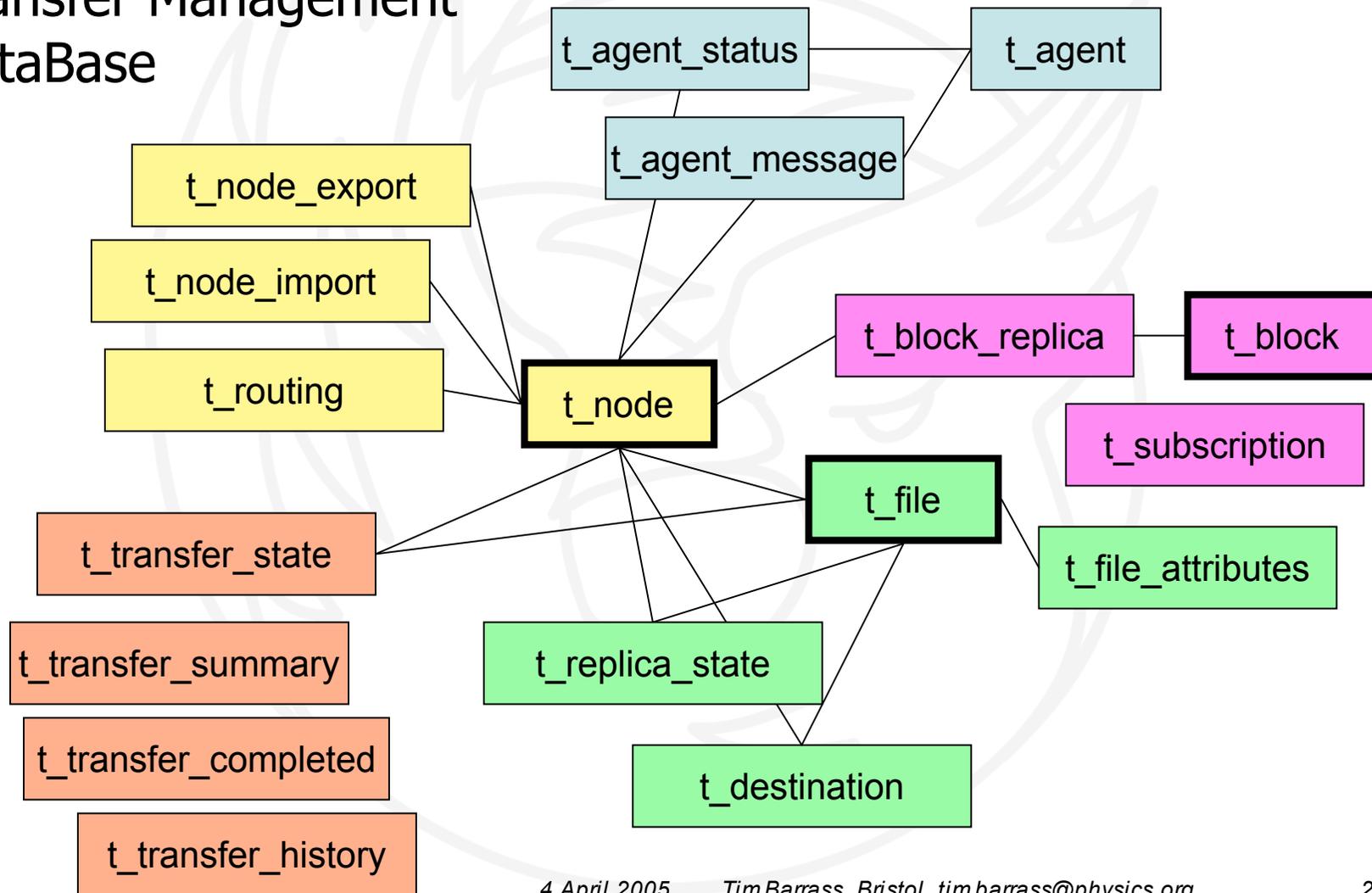
# Transfer process overview



- 1 Make subscriptions
- 2 Allocate files to destinations
- 3 Locate replicas and trigger transfers
- 4 Prioritize and mark files wanted
- 5 Make files available
- 6 Transfer file

# Blackboard deployment and agent messaging

## Transfer Management DataBase



## Introduction

- “Subscription” used to manage push and pull use cases
  - Only difference is the actor making the subscription
  - Subscription is of form “dataset:final destination”
    - Parallel senses of “requested” and “allocated” subscriptions
- Allocator agent monitors new subscriptions
  - Allocates files of a dataset to the final destination
  - Acts as a *very* simple replica manager
    - Place to start adding clever replica managers- reallocating based on global network knowledge; collaboration policy; &c
  - No transfers triggered at this stage- just allocation to final destination

## Harvesting new files

- Files placed in local buffer, available via gsiFTP/SRM
- Agent chain processes file bookkeeping data
  - Generating process creates a “drop”
    - Drop contains file indicating which files are where
    - Drop placed in “inbox” at agent at the head of injection chain
  - Drop processed by agent chain
    - Files are sized, checksummed- details added to drop
    - Files are published to some local file catalogue
    - Files are published into PhEDEx for allocation and transfer
    - File merging is also possible at this point

## On-arrival processing

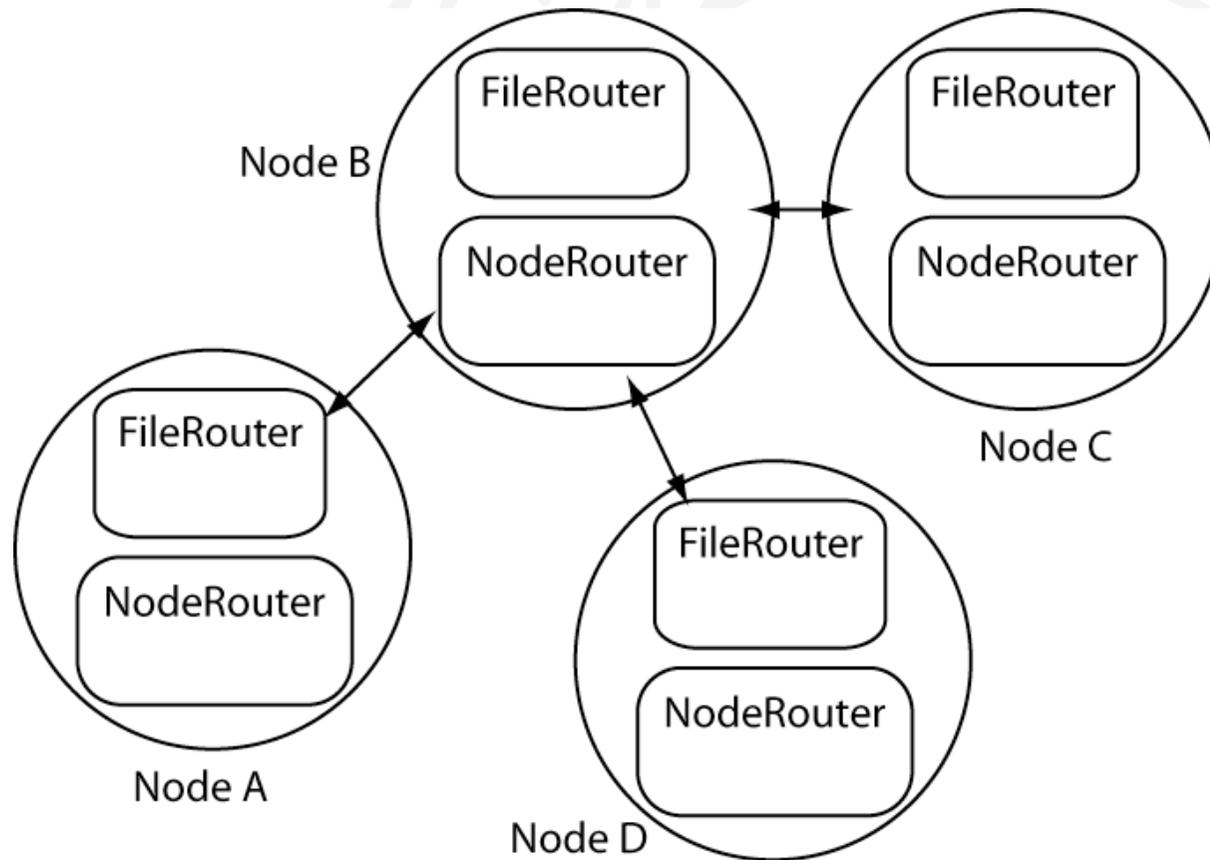
- Block monitoring agents recognize that files have arrived
  - When a complete block of files has arrived they trigger an action
- Currently used in large-scale monitoring and to link with CMS dataset publishing mechanisms
  - People want to know when a dataset is complete
  - Data published by block in PuBDB
- Also used to run analyses during DC04
  - 20 minute latency between files becoming available at CERN and results being available at T1, T2.

## Introduction

- Why do multi-hop transfers?
  - Caching at Tier 1 and “serving” to Tier 2+ reduces load on Tier 0
- We want to ensure that raw data particularly is safe on tape before deleting it on Tier 0 buffers
  - But we also need to clear the Tier 0 buffers as soon as possible!
  - Cache at Tier 1 buffer while waiting to get to tape
- We want to distribute simulated data from producing Tier 2 sites to interested Tier 1s and Tier 2+s
- Create a network overlay (in peer-to-peer terms)
  - A routed network that overlies the internet
  - Maintained by agents that act as routers
    - NodeRouters maintain route information
    - FileRouters route files from point to point toward their destination

## PhEDEx in detail: routed multi-hop transfers

# Agents involved



Routing tables

From	To	Via	Hops
A	B	B	1
B	A	A	1
A	C	B	2
A	D	B	2
...	...	...	...

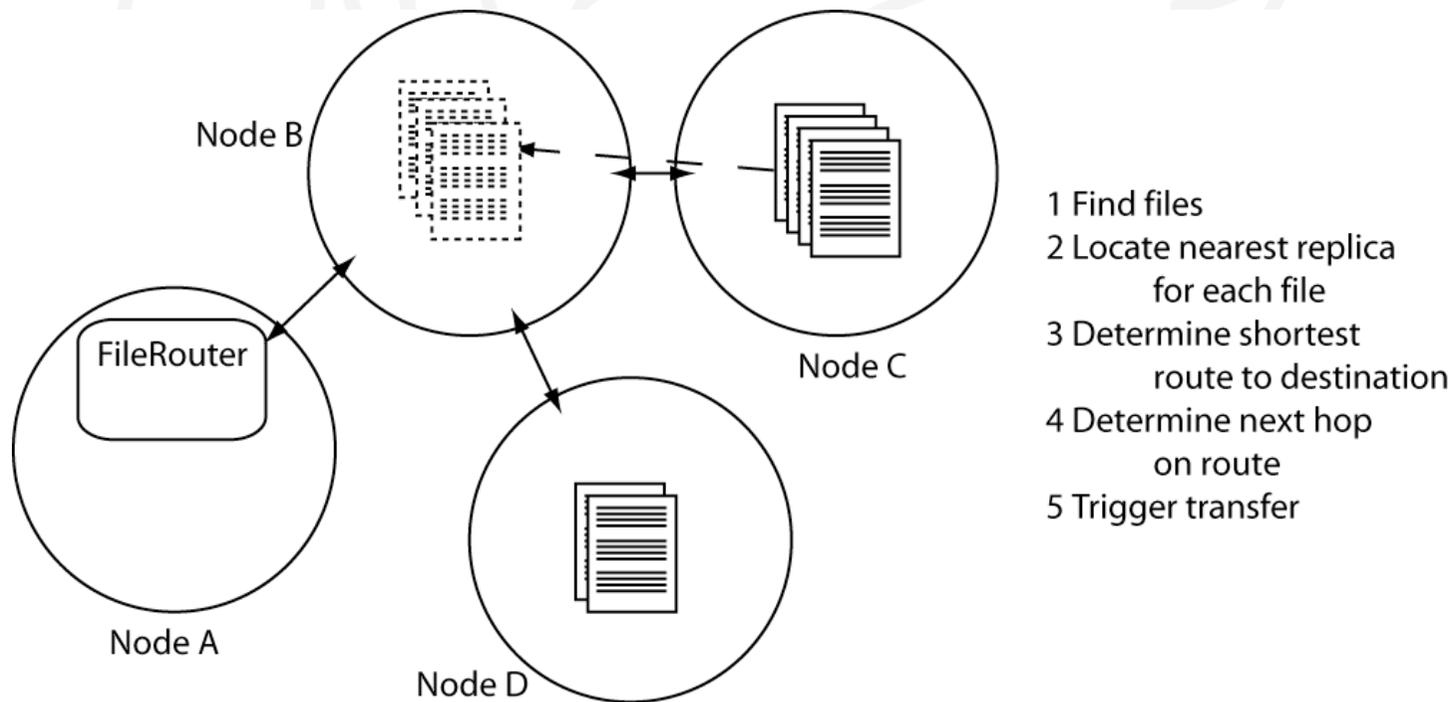
## PhEEx in detail: routed multi-hop transfers

# IP-like routing algorithm

- Routing is handled with an implementation of the Routing Internet Protocol (RIP V2, see RFC2453)
  - No message passing directly between the agents
  - Routing tables managed asynchronously in a central database
  - Routing tables contain a row for each route
    - From, to, via, hops, timestamp
- Simple distance-vector algorithm
  - Nodes are basically each 1 hop apart
  - Can “weight” hop-distance between nodes to make some routes less favourable
- Population and maintenance of routing tables handled by a NodeRouter agent
  - Associate nodes with one or more neighbours
- Routing algorithm goes as follows
  - Refresh links
    - NodeRouter updates its entry in its neighbours’ routing tables
  - Query neighbours’ routes to compare with known routes
    - Split horizon with poisoned reverse for removing cyclic routes
  - Timeout routes
    - Triggered updates- timeout everyone’s route to node via me

## File route choice

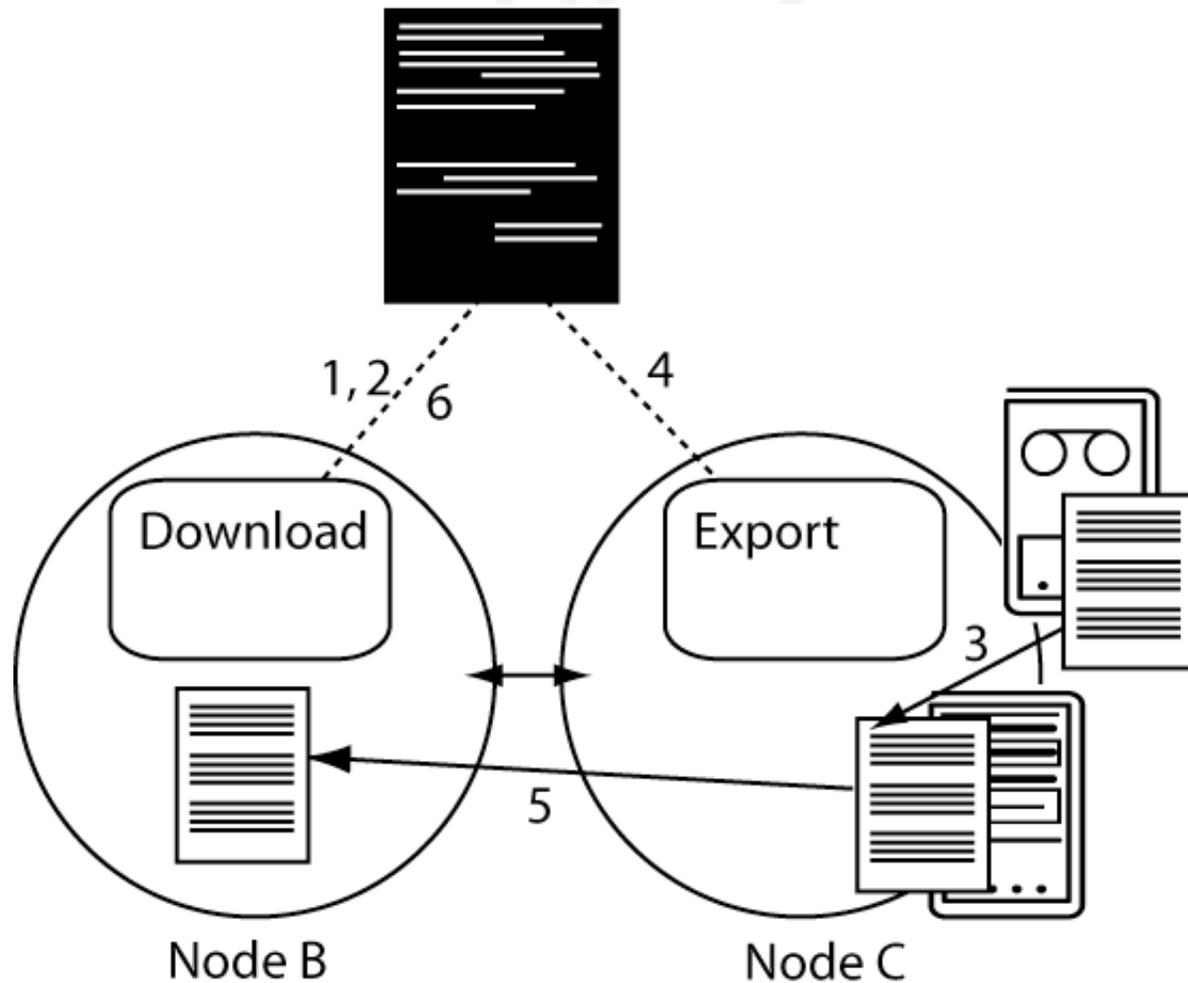
- FileRouter agent acts on behalf of a given destination node
  - Determines closest replicas, triggers point-to-point transfers
  - Dynamic route adjustment
- Here is where global and local priorities need to be resolved



## Introduction

- Experience shows that fabric and tools are unreliable
  - Tools return incorrect error codes; disk write errors (have seen problems with 1 in 1000 files); &c
- Rather than overload transfer tools with functionality, handle verification and publishing roles separate to transfer
  - BUT whole operation- transfer and publishing- must be a complete transaction
  - “Transfer” actually a multi-stage step
    - Pre-delete files if they already exist
    - Replicate file
    - Verify existence, size, (checksum) of replica
      - Delete new replica if failed
    - Publish new replica to local catalogue
    - Let PhEDEx know transfer complete

# Transfer handshake



- 1 Find transfers
- 2 Update some transfers to "wanted"
- 3 Make file available
- 4 Update to "available"
- 5 Transfer files
- 6 Update transfer as success

## Fabric and tools

- Disk resources
  - Raw NFS (globus-url-copy to local file)
  - EDG Classic SE (globus-url-copy to gsiFTP server)
  - dCache/SRM (srmcp-managed gsiFTP to dCache pool)
- Tape resources
  - Castor
    - CLI used to manage backend operations
    - srmcp and globus-url-copy for transfers
  - dCache (FZK only)
    - dccp used to stage, migrate from a local transfer buffer
  - Enstore
    - Access via dCache SRM
  - ADS (RAL only)
    - Access via dCache SRM

## PhEDEx in detail: reliable point-to-point transfer

# Issues

- Most issues fabric-related
  - Most low level components experimental or not production-hardened
    - SRMs, dCache, EDG SE ...
  - Some deployed in non-scalable configurations
    - e.g. NFS mounting disks to dCache pool nodes
- Tools very unreliable
  - Incorrect or uninformative error messages appear frequently at high load
- MSS access a serious handicap
  - Many transfers still sourced at CERN
  - CERN Castor stager is unable to cope with all demands
    - PhEDEx plays very fair, keeping within request limits and ordering requests by tpe
    - Other users don't- crippling the stager
    - With exclusive use we can get 25MBps Castor tape->remote disk sustained for days
  - Enstore SRM access appears much more performant
- Scalability problems not all PhEDEx related
  - Improvements made to database have enabled increasing volume
  - Main problem is keeping in touch with the O(3) people at each site involved in deploying fabric, adminstrating &c
  - Configuration and scalability problems seen with dCache as well as Castor

# Summary

- Dataset-scale transfer management
  - Possible to trigger dataset-level transfers then leave the system to ensure they reach their destination
    - Fire and forget?
    - New files automatically harvested and placed in distribution
    - Files can be processed automatically on arrival at destination
- Reliable multi-hop transfers
  - Key aim is to enable the safe but rapid clearing of buffer space
  - CMS- and HEP- use cases for transfer more sophisticated than simple point-to-point.
  - Routing of files through system dynamic
  - Choice of closest replica to destination
- Robust point-to-point transfers
  - Tools are typically unreliable
    - Where do activities like checksumming, verification belong?
  - Pre-delete, transfer, verify (and post-delete), publish new replica information, complete transfer



Future work on PhEDEx

## Introduction

- PhEDEx beginning to face scaling problems- as expected
  - Many problems solved by using non-naïve database deployments
  - Currently exploring other technologies as means of distributing information
    - Peer-to-peer technologies
    - Agent framework standards
  - Major problems still in fabric management
    - Maintaining contact with all local administrators to subtly modify e.g. transfer parameters is not scalable
- Some aspects of these technologies already in PhEDEx
  - Routed network effectively a peer-to-peer style network overlay
  - Download of dataset parts (files) from nearest/lowest cost replicas similar to p2p filesharing apps
    - Not currently as sophisticated
  - Agents trivially collaborate to solve defined problems

# Contractual file routing

- File routing agent requests supply of a file
  - Creates a formal routing request with a certain time validity
  - Routing agents at nodes with replicas estimate “cost” of transfer, then tender for transfer with this information
    - Indicating which is the next node in the route
  - Intermediate nodes successively make further tenders for their hops
    - Until finally next node == destination
  - Routing agent chooses between them based on total cost of each route
- Need to handle failure of routes
  - Timeout whole routing offer in case a node in route fails
  - Cost needs to include a reasonable estimate of ability of node to fulfill request
    - A node may already have a large backlog of transfers to handle ...
  - Also- don't want to oscillate through routes on continued failures
- Format for request-tender resolution quite well understood in many places
  - Standardized and implemented in many agent frameworks
    - e.g. contract-net decision flow
    - See for example the FIPA contract-net description
      - <http://www.fipa.org/specs/fipa00029/XC00029F.html>

# Peer-to-peer data location

- Aimed at abstracting parts of the TMDB
  - Develop as a new network overlay
  - Use for data block location
- Use Kademlia algorithm
  - Nodes represented by a hash of their name, other info
  - Use same hash function to hash e.g. data block name
  - Map location information onto N nodes represented by hashes (algorithmically) close to the data block name hash
    - e.g. not topologically close
  - Those N nodes are then responsible for storing and maintaining information about that data block
  - Information is timed out and refreshed
    - Nodes will find that they are no longer one of the closest and can therefore drop information
- Exploratory work underway with Kenosis
  - Provides infrastructure for node discovery
- Issues
  - Threshold minimum number of nodes needed to make this worthwhile?

Future work on PhEDEX

## Semi-autonomy

- Agents are best placed to monitor and respond proactively to local conditions
  - They control many small scale tuning parameters
    - TCP window sizes, block sizes, number of parallel transfers ...
  - Can sense changes in actual achieved transfer rates and modify their demands of particular transfer links
    - Set goals within certain limits
    - React proactively when goals are met, or when goals are no longer met
  - Message human manager when things begin to go badly awry
- Some of this can also be pushed lower, to more intelligent transport protocols

## Testing new technologies

- Already exploring the use of dedicated links and hardware
  - 2005 LCG Robust Transfer Service Challenge
    - Use of Starlight 10Gb point-to-point links
    - 10 node dual Itanium cluster at CERN
    - Similar dedicated hardware at remote sites
  - Achieves good rates (500MBps for a few hours overall, 80MBps on PhEDEx links to tape)
- Maybe examine other transport protocols
  - e.g. Bulk File Transfer protocol
  - Also requires supportive transfer tools!
    - New command line interface tools easy to incorporate into PhEDEx
    - But would need them incorporated into e.g. srmcp for use with SRM services for manageable use
- Sophisticated testing possible
  - We can fake transfer operations
  - Do dry runs
  - Remove any component and replace with simulated behaviour...
  - Excellent basis for exercising e.g. new storage tech at a single site

## Future work on PhEDEx

# Policy and priority

- The collaboration has goals that may need to be resolved with local goals
  - Collaboration has transfers that are essential- raw data to T1 MSS
  - Physics groups/individuals have preferred datasets
    - And are tied in some sense to locations
  - Individual sites want to prioritize datasets for which they are the destination
    - But also provide buffer space for through-transfers
- Looking at mechanisms to manage policy and priority
  - In technical terms policy => some scheduling algorithm and priority => hard and soft deadlines
  - Possible initial policies include fairshare, ...
  - High priority transfers map onto hard, sooner deadlines
  - Low priority transfers map onto soft, later deadlines
- Policy implemented at FileRouter, FileDownload, FileExport agent levels
  - Distributed, not centrally enforced
  - Need to define how we express priorities and policies in a reconfigurable way

## Summary

- PhEDEx now approaching a stable system
  - Always focused on production- delivering data to the experiment
  - New additions are proven before inclusion in the system
    - No “fall back” to a last known good system
- Becoming a good environment for research into new techniques in specific areas
  - Driven to explore novel solutions to problems to maintain scalability
- Current and proposed research into
  - Contractual file routing a la agent systems
  - Peer-to-peer sharing for data location
  - Testing new hardware and other technologies
  - Management of policy and priority to ensure collaboration requirements are met

# Summary

- PhEDEx is CMS' production data file transfer management system
  - Maturing now
  - Designed to make the management of large-scale transfers simple
  - Able to handle sustained TB a day transfers
- Distributed system
  - Only TMDB is currently a single point of failure
  - Transfers can continue in face of failure of nodes in network
  - Robust in the face of most local management activities
- Plenty of ongoing work
  - Into mechanisms to enable more scalable file routing, data location, and management of collaboration and local policy
- Actively seeking discourse with other groups
  - Also more than happy to help people try parts out if they wish

# Links

- PhEDEx and CMS
  - <http://cms-project-phedex.web.cern.ch/cms-project-phedex/>
  - [cms-phedex-developers@cern.ch](mailto:cms-phedex-developers@cern.ch) : feel free to subscribe!
  - CMS Computing model  
[http://www.gridpp.ac.uk/eb/ComputingModels/cms\\_computing\\_model.pdf](http://www.gridpp.ac.uk/eb/ComputingModels/cms_computing_model.pdf)
- Agent frameworks
  - JADE <http://jade.tilab.com/>
  - DiaMONDs <http://diamonds.cacr.caltech.edu/>
  - FIPA <http://www.fipa.org>
- Peer-to-peer
  - Kademlia <http://citeseer.ist.psu.edu/529075.html>
  - Kenosis <http://sourceforge.net/projects/kenosis>
- Autonomic computing
  - <http://www.research.ibm.com/autonomic/>
- General agents and blackboards
  - Where should complexity go?  
<http://www.cs.bath.ac.uk/~jjb/ftp/wrac01.pdf>
  - Agents and blackboards <http://dancorkill.home.comcast.net/pubs/>