dCache at FZK/GridKa

compiled for the preGDB meeting at
7 November @ CERN

Storage hardware plans

- dCache for disk pool management
- separate pool spaces for each VO (D1T1, D0T1, D1T0)
- TSM for tape management
- possibly separated dpm instances for disk-only and tape backed

2006

- 4 Admin nodes (HA types)
  - head node (pool manager jvm, admin jvm, http jvm, billing jvm)
  - pnfs (jvm, pnfs postgres db, companion postgres db, srm postgres db)
  - srm node (jvm)
  - spare
- 10 gridftp door servers
- 33 disk pool servers

2007/2008

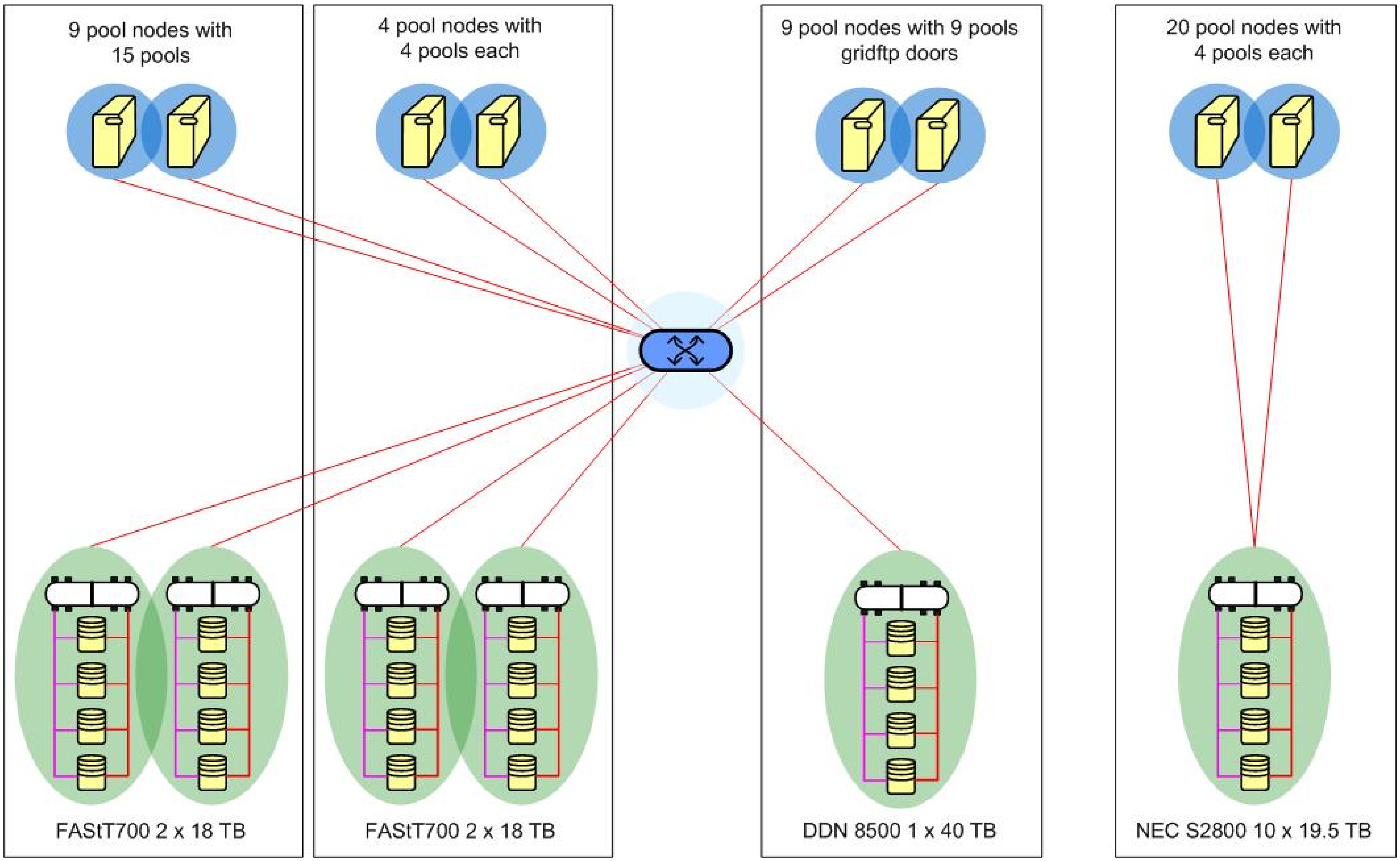
- 5 Admin nodes (HA types)
  - head node (pool manager jvm, admin jvm, http jvm)
  - pnfs (jvm, pnfs postgres db, companion postgres db)
  - srm node (jvm, srm postgress db)
  - billing (jvm, billing postgres db)
  - spare
- jvw1 gridftp doors capacity: 20 Gbit, 2 MB/stream
- disk pool servers capacity (per cpu core): 2 streams, 0.05 Gbit
- for FZK this means (4 GBmem and 1 Gb per server)
  - 20 gftp servers
  - 100 pool nodes (100 Gb and 10000 streams)

Slide 3

jvw1 Use the rough estimate of 5000 cpu cores in 2008 for LHC

Jos van Wezel, 11/6/2006

pooltypes by hardware



Pool types by function

A (T1D0):

- Input Write Buffer for raw data (select via routing)
- Sizes according to expected input stream
- separate set of pools
- dual location

B (T1D1):

- Input and output buffer for T1 and T2 inter-traffic (e.g. AOD from T2, replicas to T1)
- pools are sharing pool hosts with type C and D

C (T0D1):

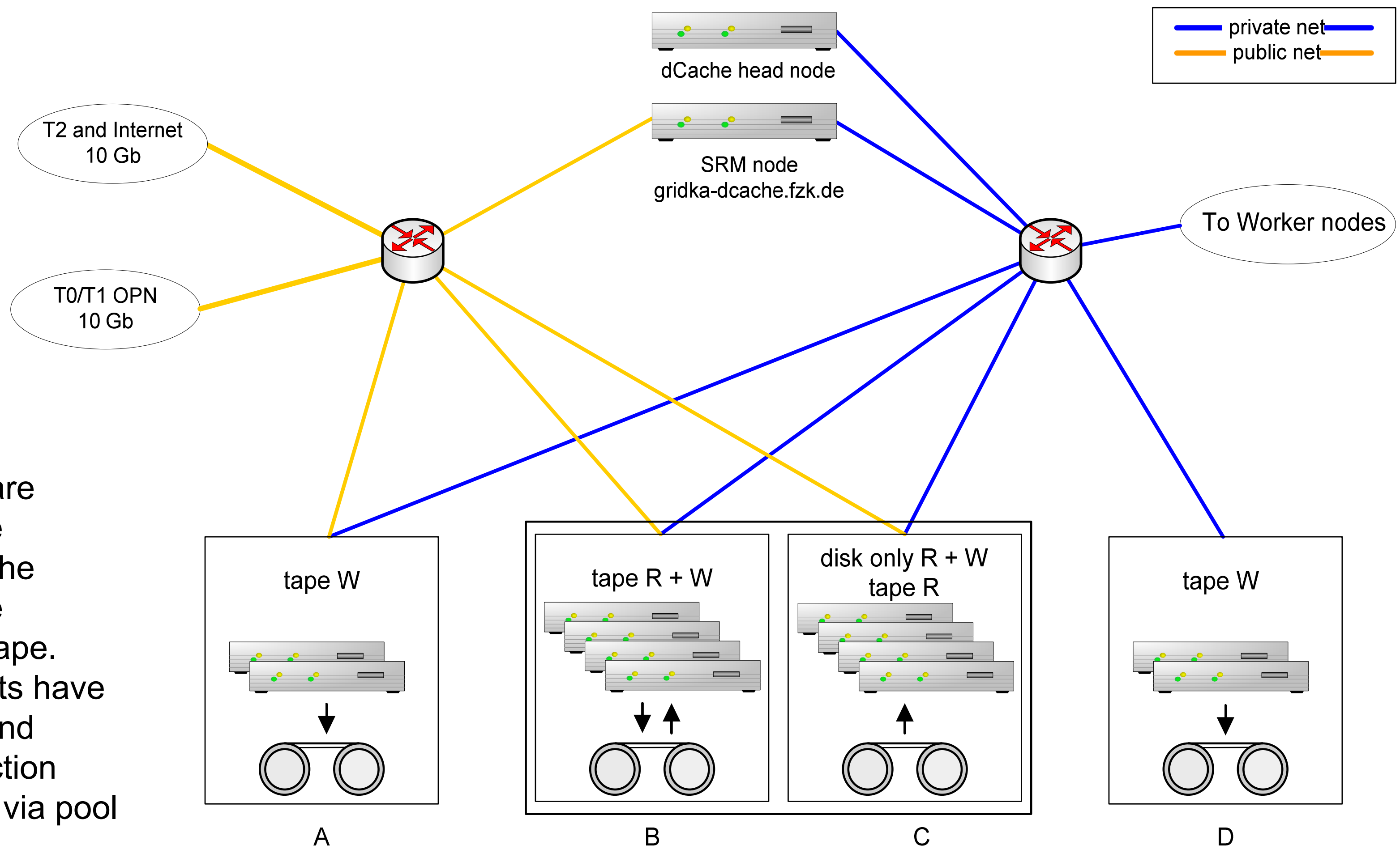
- Disk only
- Selected via path in dCache. For other paths this pool is like type B

D (T1D0):

- locally produced data for which we are custodian. Amount unclear but low tape demand.

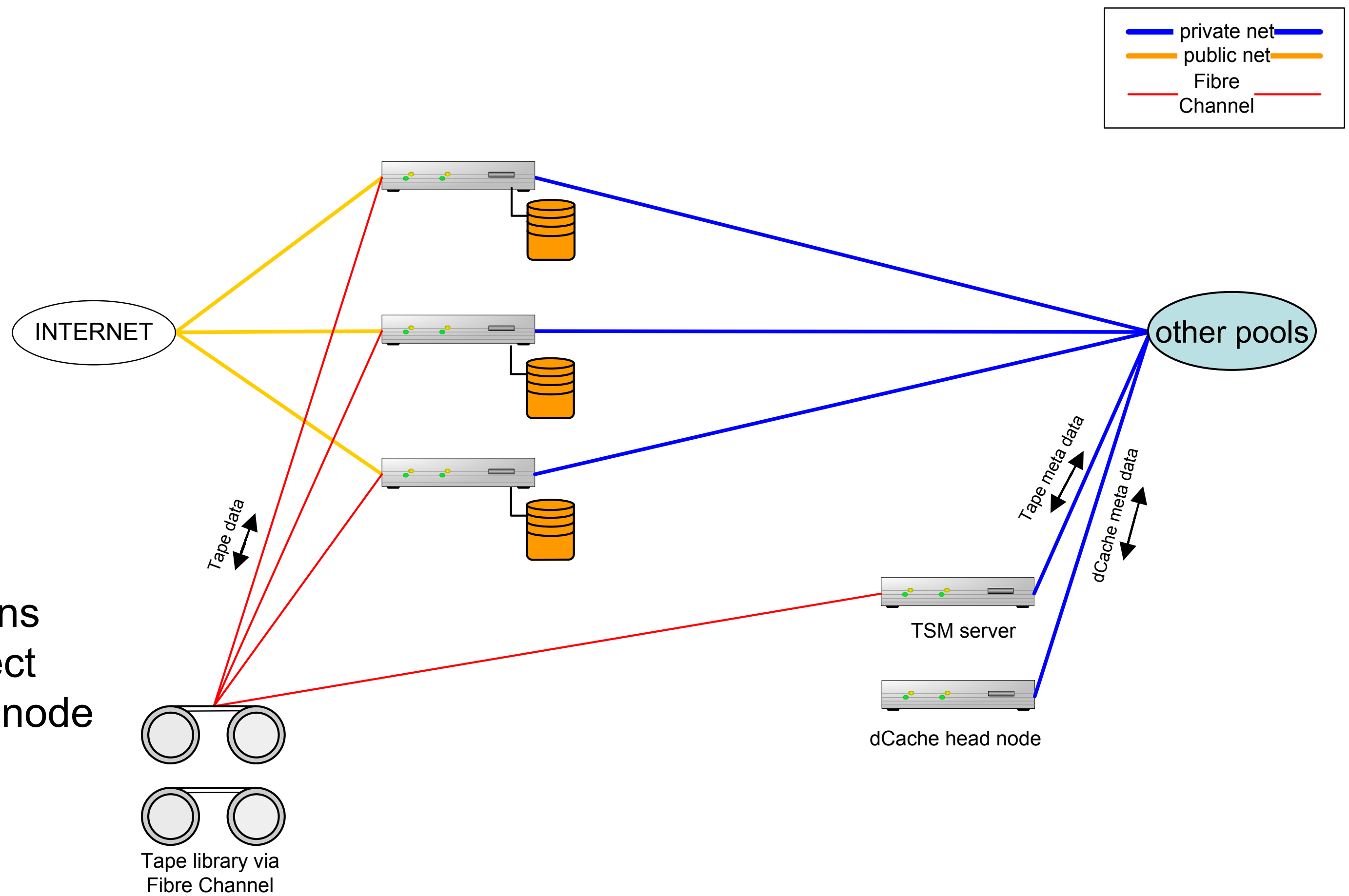
Pool host sharing: to better utilize installed hardware
Pools that receive data (write pools) have FC disks and a direct connection to tape via FC
Pools that send data (read pools) have SATA disks and connect to tape via tape storage groups of 4 to 5 pool nodes

All pool types (D0T1, D1T1, D1T0)



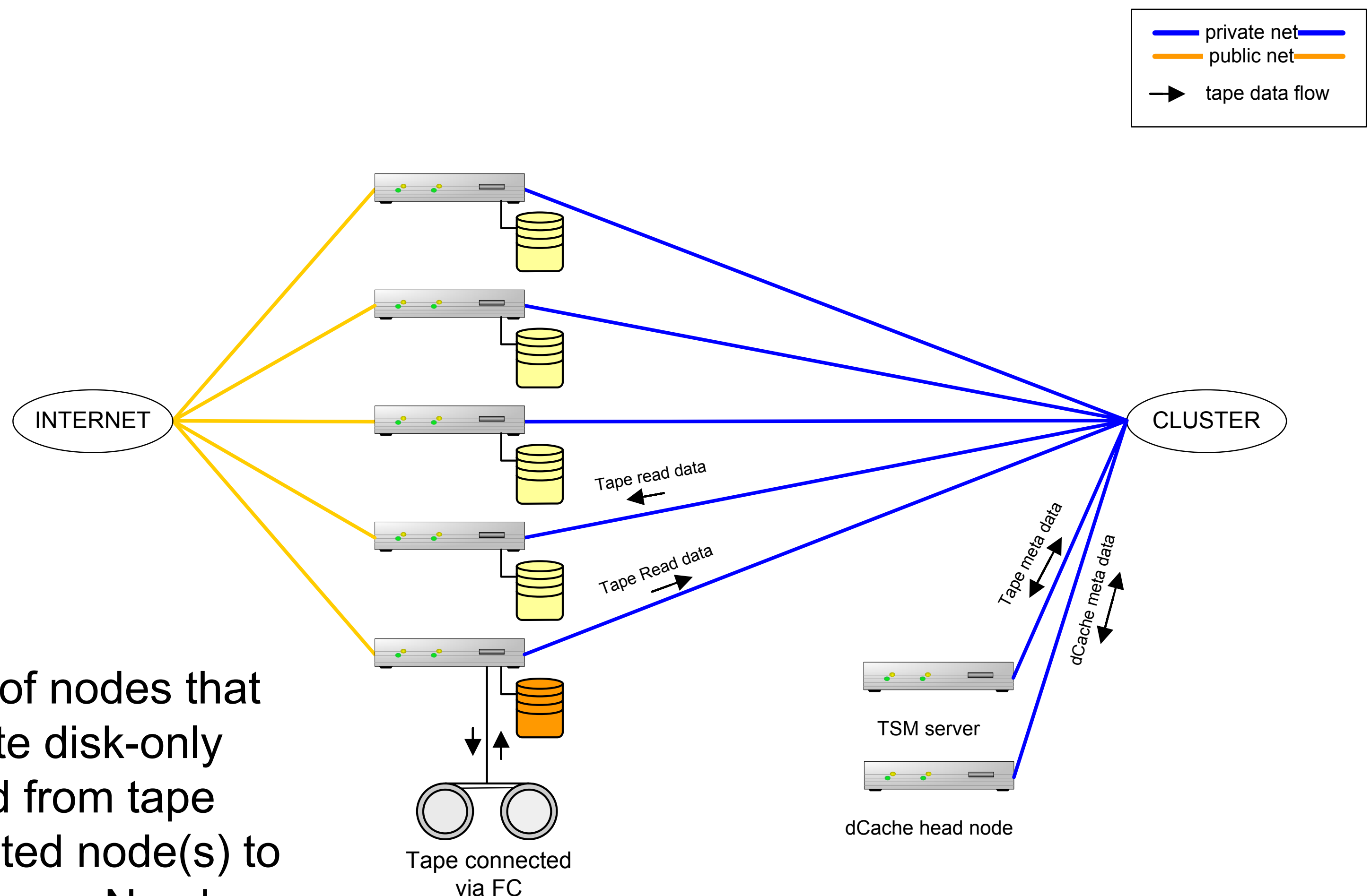
- Types B,C,D are in fact the same machines with the exception of the host writing to tape.
- Tape write hosts have FC disk pools and FC tape connection
- Tape buffering via pool

Input buffer pool (A)



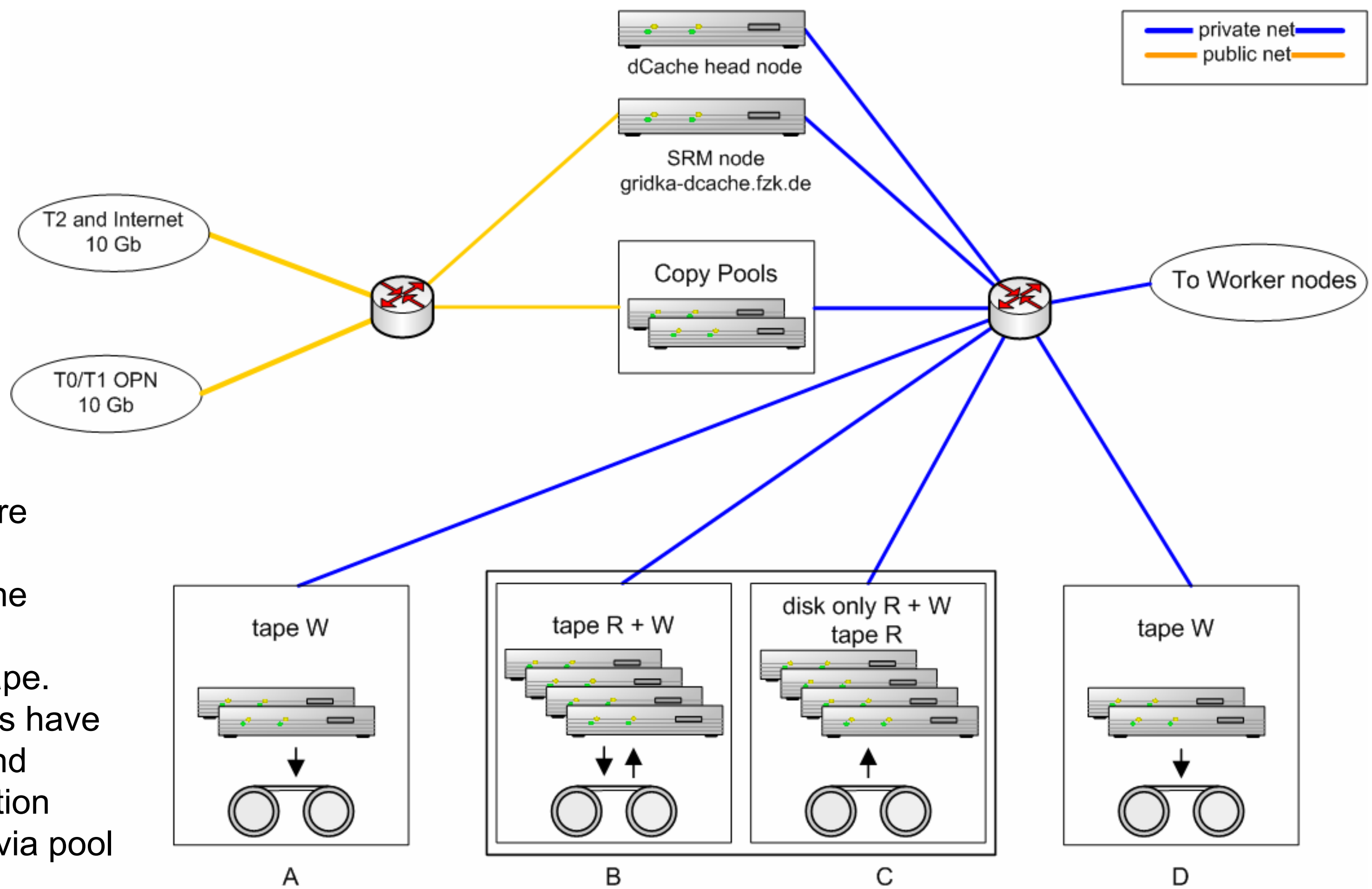
- Dedicated pool to accept T0 to T1 data
- WAN buffer
- 48 hrs buffer
- Redundant in 2 locations
- Redundant tape connect
- Possibly at 10Gbit per node
- No pinning allowed

Pool node set B,C,D



- Group of nodes that read/write disk-only and read from tape
- Dedicated node(s) to write to tape. Number can be adapted (for the tape0 tape1 transition?)

All pool types (D0T1, D1T1, D1T0) external connection via pp copy

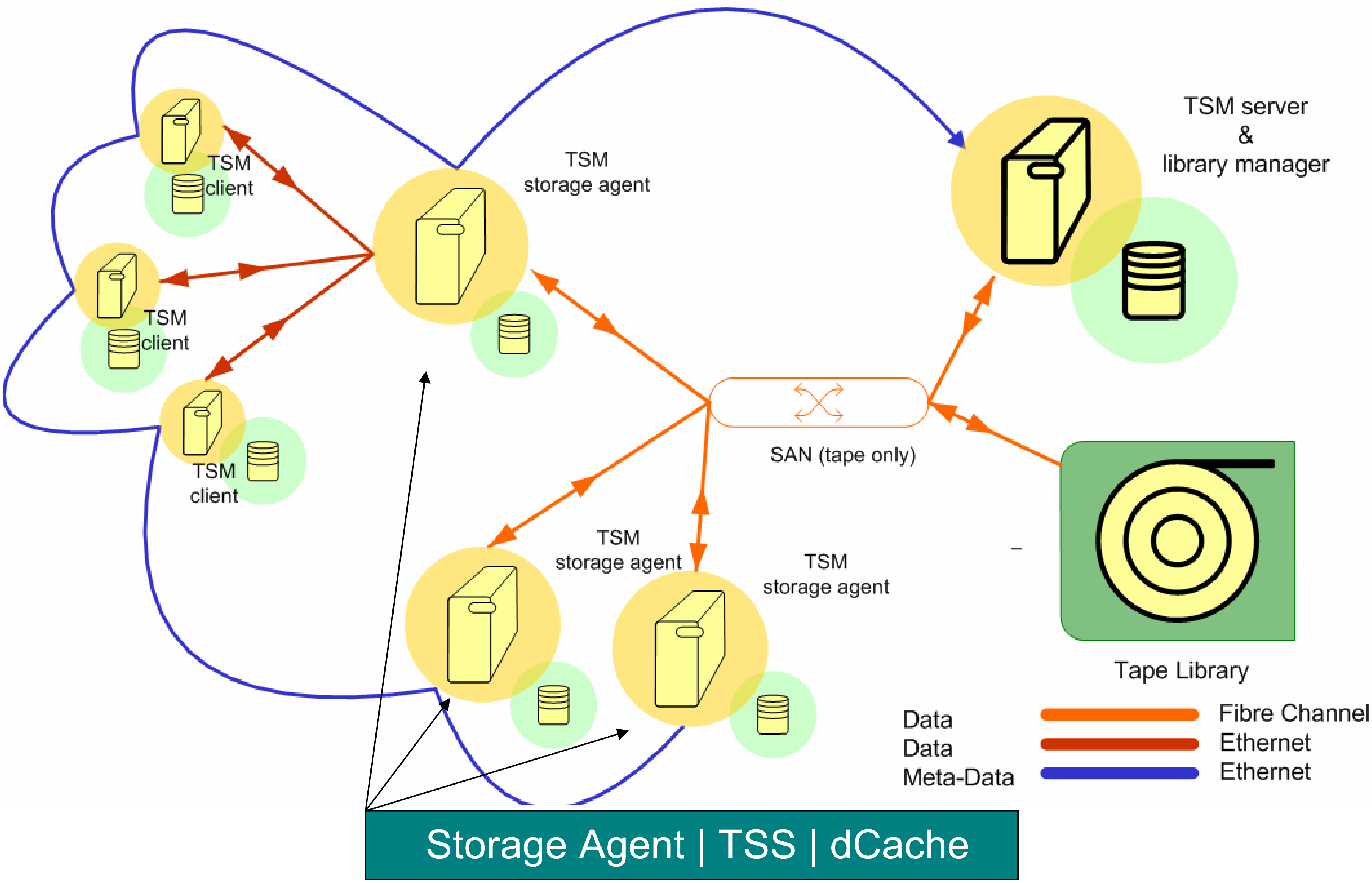


- Types B,C,D are in fact the same machines with the exception of the host writing to tape.
-Tape write hosts have FC disk pools and FC tape connection
-Tape buffering via pool

WAN connect via separate PP pool set

- Copy through function of dCache
- Allows incoming data to be forwarded to a different (internal) pool.
  - available in version 1.7
  - The reverse is likely also possible
- Traffic from extern is handled with a dedicated set of nodes
  - select on the basis of the path if data goes to tape also (or it just passes on)
- Pool2pool setup (F) compared with ‘all connected’ (A)
  - F: reduced number of hosts with external connection
  - F: hosts cannot do much else and must be maintained. Need ~10 forwarder hosts (500 MB/s) not counting redundancy: 22k euro
  - A: with ~100 pool nodes extra ports needed: 25k euro
- It looks like ‘all connected’ is the better option

TSM (meta)data flow with storage agents



TSM Session Server
dcache to tape interface

- Uses the dCache pools as tape buffer
- Interfaces directly with TSM via its API
  - the API libs come with the TSM software
- Single executable, documentation 'tss –-help'
- Fan out for all dpm to tape activities
  - single session to the TSM server
  - multiple tape flush/retrieve/rename/log/queries
- Runs on the TSM clients, storage agent or on the server proper
- Plug-in replacement for the TSM backend that comes with dCache
- Sends different type of data to different tape sets
  - if known from dcache 'tag'
  - groups data that are likely to be recalled together
- Queues multiple requests (no state is kept, dpm must re-queue if needed)
  - support from DPM on recall needed
  - if possible also for stores
- Allows to store an exact image of the global name space on tape
  - store the 'site file name'
  - decoupling of disk pool manager and tape backend
  - needs 'rename' support of the dpm

- Before SC4: max 40 MB/s on 8 drives and 1 server
- During SC4: max 250 MB/s on 8 drives and 8 STAs
  - working number for planning is 30 MB/s per drive

Enhancements

Reading

- Sort recall order on tape file sequence
  - needs support of the storage manager

Writing

- Improve throughput (LTO3/LTO4)
  - decoupling reads and writes
  - Include sizing estimates on write
  - throttle or stop writes based on node IO load

Support for xrootd

- can use the same interface
- planned for early 2007

10 Gb networking

- may use the Ethernet again for tape operations

Improved Scheduling

- TSS to TSS communication needed?
- support from storage manager needed

Implement Disk staging?

Unresolved (dCache oriented)

- Glue → SRM → Path / Token?
  - need to have fixed paths
- Disk space consolidation
  - we now have pools all over the place
  - deleting data, moving data
- Disk in front of tape
  - how is this to be implemented
- 10 Gbit on servers
  - that's the complete LCG MOU throughput of GridKa on 1 single machine!! (in theory)
- Copy through mode (P2Pcopy)
  - to provide buffering (WAN/tape)
- Firewall issues
  - third party put mode goes through the NAT if you have a private net
- Stability issues
  - seems to be resolved
- tcp/ip settings (buffer sizes, keepalive etc)
- 2 dcache instances?: tape and disk-only

Site considerations

- classes implementation should reflect actual storage costs (€€)
  - quality x access time x costs = storage class (SA?)
- default storage class should be most expensive
  - (e.g. reduce fair share of VO)
- should be convertible with little admin effort (i.e. no data moves)
  - make room on demand (D1T1 to D1T0 and v.v.)
  - preset specific processing patterns
- D1T1 how large is d in T to D
  - see above: need tools to convert D1T0 to D1T1 space and back
- what path to use for a given class: is storage space = path?
  - FZK uses
    - tape backed: $VoName/
    - disk-only: $VoName/diskonly/
- available space (disk and tape) must be reported correctly
  - 'gap' space in dcache
  - reserved space on disk (could be fs specific)
  - how large is tape space
- pinning on disk is left completely to the user/experiment
  - expect users/experiments to request status (pinned vs available space)
- D1T0 to D1T1 transition will consume more tape drives