

The Pouzin Society

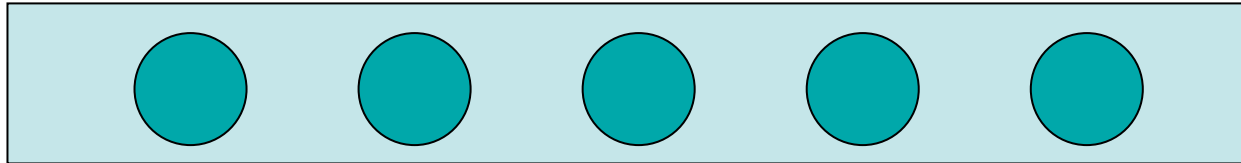
DAFs and Management in RINA

IRATI Workshop

Barcelona, Spain

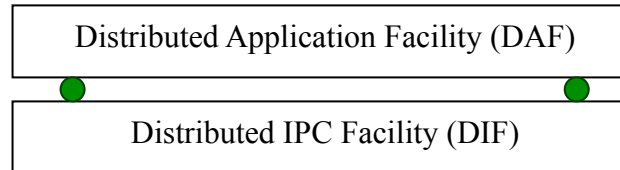
John Day and Eleni Trouva

A DAF consists of two or more DAPs

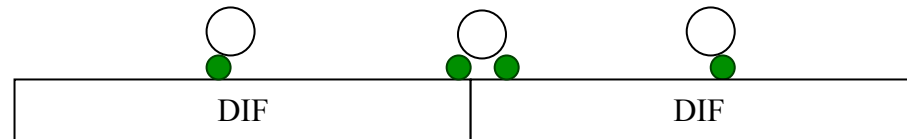


- Notice that everything is a DAF. An Application that does not use IPC has no output and hence does nothing.
- A DAF in which all DAPs are of the same type is *homogeneous*.
- A DAF with DAPs of different types is *heterogeneous*.
- A new Application Process joining a DAF must enroll.
 - It works just like DIFs, actually DIFs work just like DAFs.
- The DAF may assign the member DAPs a synonym with scope limited to the DAF and structured to facilitate its use within the DAF.

DAFs Operate Over A DIF

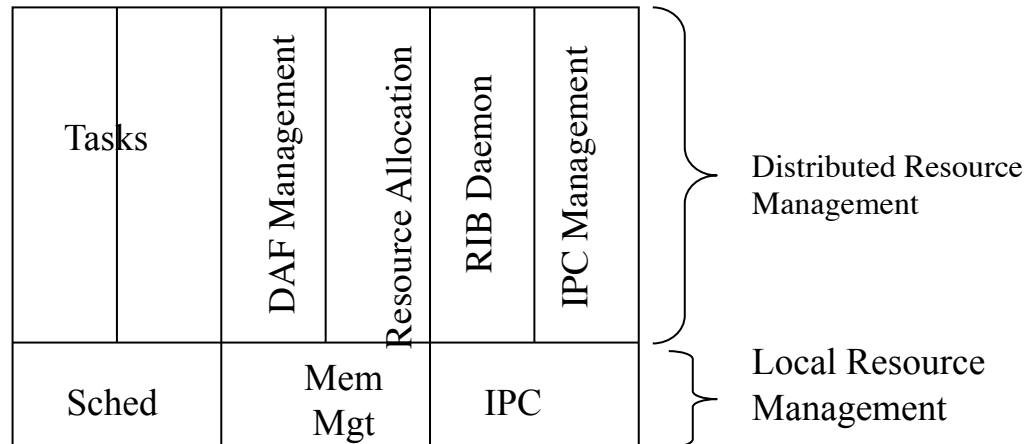


Can a DAF span more than One DIF?



- There seems to be no architectural reason why not.
- This requires at least one DAP relaying which could allow information to leak between domains.
- Rules can be made but are hard to enforce, but these are not as strong as it being enforced by the structure.

A DAP Consists of

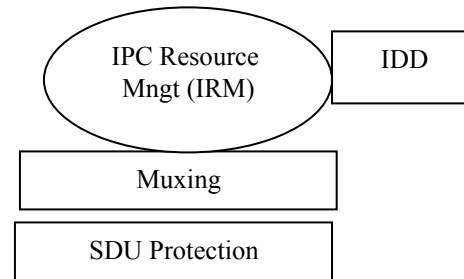


- This requires considerably more exploration.
- Conjecture: In general the Tasks do not use IPC, but the RIB Daemon makes the information available that the tasks need.
 - IOW, the function of a distributed application is reduced to a local programming problem.
 - Not only is there only one application protocol but there is only one user of that application protocol?

DAP Infrastructure

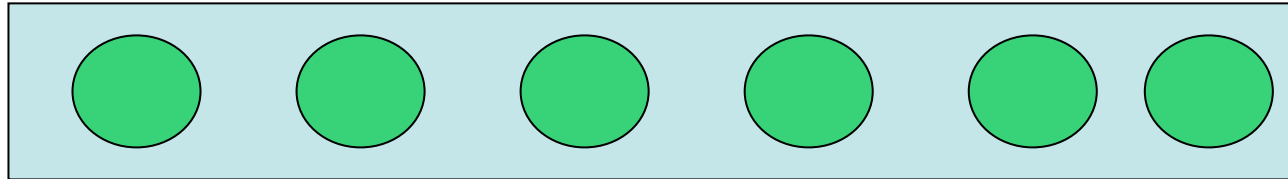
- **DAF Management** is the local task involved in the management of the DAF as a whole. It can range in complexity from a simple agent to a full participant in the management. (more on this later)
- **Task Scheduling** - is the local task that coordinates with its peers the work of the DAF. (In a DIF, this is generally relates to routing and QoS.)
- **RIB Daemon** - is the local task that ensures replicated information in the RIB is updated as required and services requests for information from the Tasks of the DAF. (In a DIF, this is a generalization of combining routing update and event management.)
- **IPC Management** - IPC Management manages the DAP's use of the underlying DIF to communicate with its members. (There is much more to say about this)

IPC Management



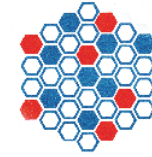
- In the IPC Model, there was a function that was used to choose which DIF to use. This is it.
- IPC Management is the part of a DAP that manages the use of the supporting DIF.
 - SDU Protection and Multiplexing are the same as in DIFs.
 - The IPC Resource Manager (IRM) does the actual management
 - The Inter-DIF-Directory (IDD) is used to find applications that may be on DIFs that this DAP does not have direct access to.

Naming Considerations for DAFs



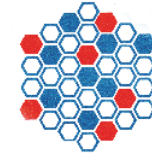
- The members of a DAF cooperate to perform a set of functions. Hence, they may have a shared schema that describes the information they use.
 - And policies governing replication, ACID, authoritative value, etc.
- This schema may or may not be made visible to the users of the DAF.
- One of the roles of the RIB Daemon is to maintain the mapping between this schema and how to access the information, i.e. where in the DAF this information resides.
 - Hence, synonyms may be assigned to facilitate this, e.g. DHTs, LC
- The Tasks of the DAF use this schema to access the information required to perform the functions of the DAF.
 - The schema made visible by the tasks may be different than the schema used within the RIB.

Very Interesting . . .



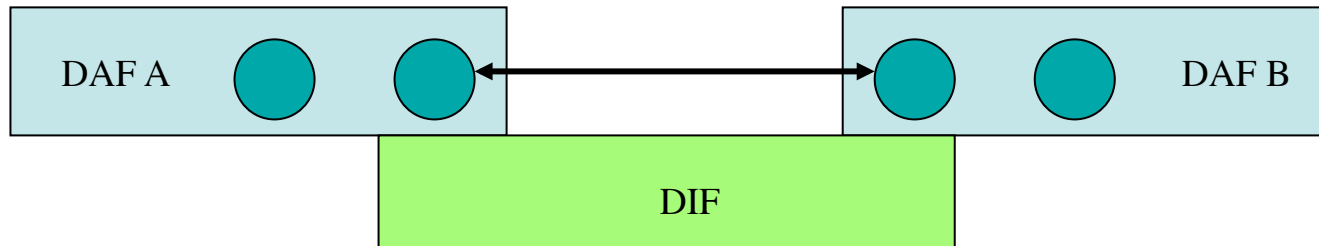
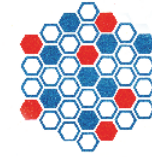
- How much in common DAFs and DIFs are:
 - SDU Protection
 - Multiplexing
 - RIB Daemon
 - Enrollment
 - Addresses (Synonyms)
- Ignoring differences of policies, not much is unique to DIFs:
 - Flow Allocator
 - Delimiting
 - Error and Flow Control
 - (Relaying)
- Like I said, Very Interesting. . .

Can DAFs Use DAFs?



- Yes, of course. But there are two forms it can take.
 - Invoke
 - RPC
- In general, a DAF provides some function (or set of functions) and will provide the result of that function to the member of the DAF that invoked.
 - Assume DAF B provides $f(x)$ and it is invoked by a member of DAF A
 - B returns the result to the member that made the request. The fact that it was a distributed computation is not visible to A, is termed *asymmetric*.
 - A rare form of DAF, where performing $f(x)$ by one user may result in $f^{-1}(x)$ being performed elsewhere, is termed *symmetric*, e.g. a DIF.

Can DAFs Use DAFs? (cont)

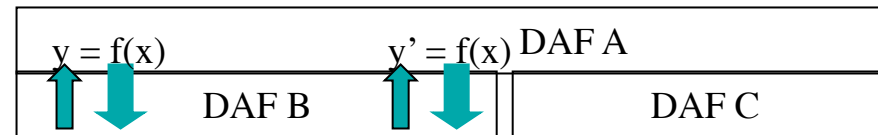


- The Remote Procedure Call form:
 - A DAP, **a**, in DAF **A** opens a connection to a DAP, **b**, in DAF **B**, (which includes authenticating) and sends $f(x)$.
 - **a** and **b** must be in two DAFs at the same time.
 - This could constitute any number of security problems.
 - Information available to **a** as a member of **A** may not be shared with **B**. Major assumptions have to be made about the veracity of **a**.
 - The previous method provided more structural isolation.
 - Note that $f(x)$ is not enrollment. For this sort of service, there are a number of possibilities: distinct DAN, distinct AE in a DAP, or distinct DAPs to provide the function and isolate it from the rest of the DAF.

Can DAFs Use DAFs? (cont)



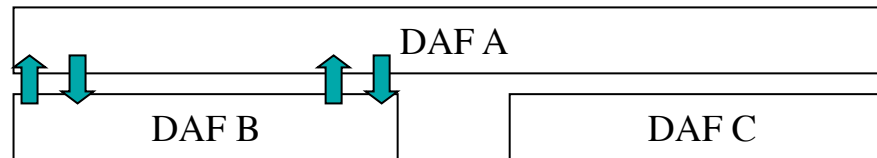
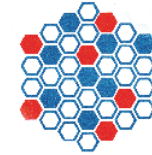
- Do all members of A have access to the same supporting DAFs?
 - For a homogeneous DAF, yes. For a heterogeneous DAF, perhaps not.



where y may not equal y'

- If one member of A invokes $f(x)$, the result might not be the same if another member of A invokes $f(x)$

Can DAFs Use DAFs? (still cont)

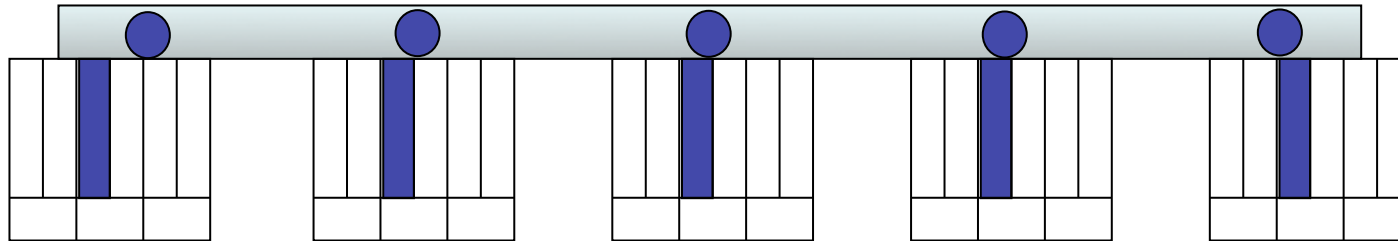


- A DIF is a special case such that when $f(x)$ is invoked, $f^{-1}(x)$ is invoked elsewhere, i.e. symmetric.
 - Are there other forms of symmetric DAFs where $f(x)$ causes action at a distance where f , f^{-1} , or even g are invoked?
 - Yes, email, various “messaging” schemes or some delay tolerant networks
- Ultimately, it would seem that a DAF has at least one supporting DIF for sharing information among its members.
 - Is there an example that proves this statement wrong?

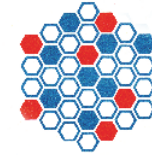
Conjectures

- Peer to Peer [sic] systems are asymmetric homogeneous DAFs, where the RIB Daemon maintains a schema to locate information at one or more members of the DAF and then transfer the information to the requesting DAF member.
- Email is a symmetric DAF that stores a message with the user of another member. The message may be retrieved at some point by another member of this DAF or by another DAF.
 - Mail could be a DIF if there is an upper bound on how long a message will be held before pick up.
- Content-centric networking is simply a distributed database DAF.
- Others?

Management DAFs



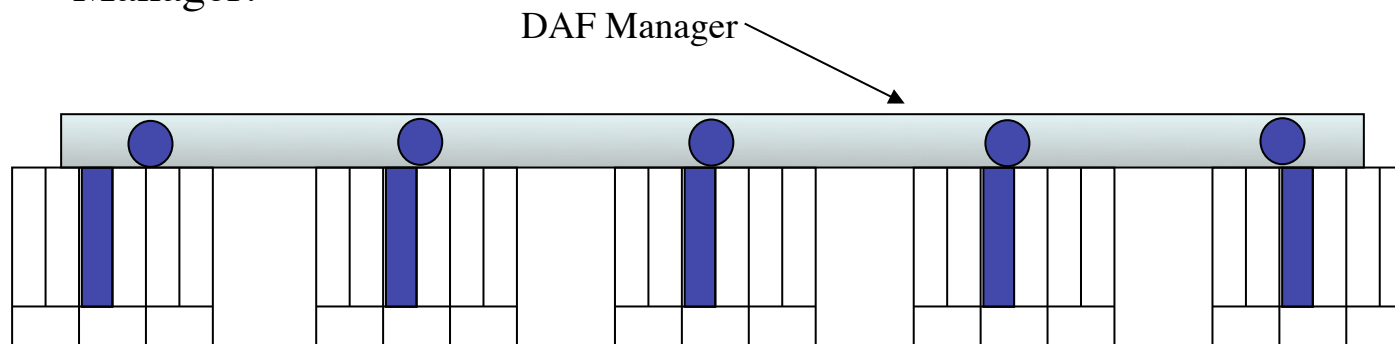
An Important Class of DAFs: DAF Management Systems



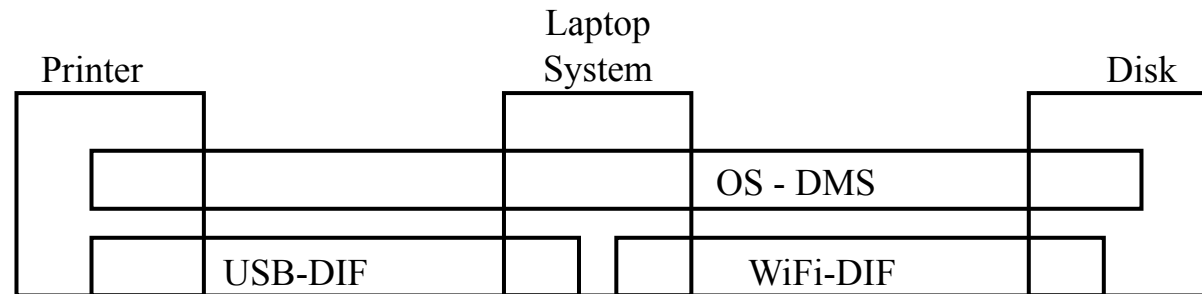
- There are four major kinds of distributed management systems (DMS):
 - Operating System
 - Network Management
 - Distributed Applications
 - Name Space Management
- There were the beginnings of progress in this area in the late 80s.
- However, thanks to the SNMP debacle of the early 90s,
 - The IETF were played for suckers and took the bait
- It pretty much reverted to the primitive state of 70s with ad hoc, largely, proprietary solutions and kludges,
- Effectively aimed at keeping account control and using management as a barrier to entry.

DAF Management Systems

- There is a commonality to their structure and
- A range in their complexity from distributed to centralized
- Each DAF/DIF has a DAF Management Task. These constitute data collection and autonomic functions, what IEEE calls layer management.
- The DAF Manager can be considered the nervous system of a DAF.
 - A DAF Manager might manage more than one DAF or
 - In a degenerate case, the DAF Management Tasks might constitute the DAF Manager.

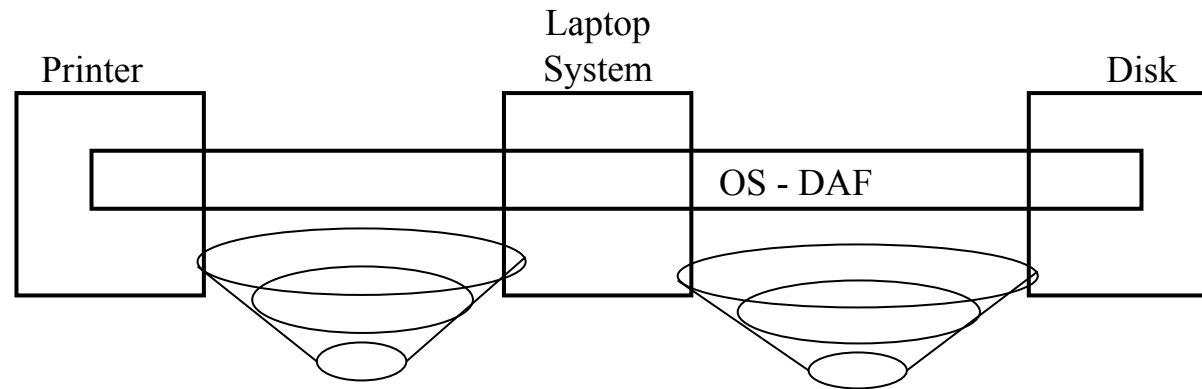


Application and Operating System DMS



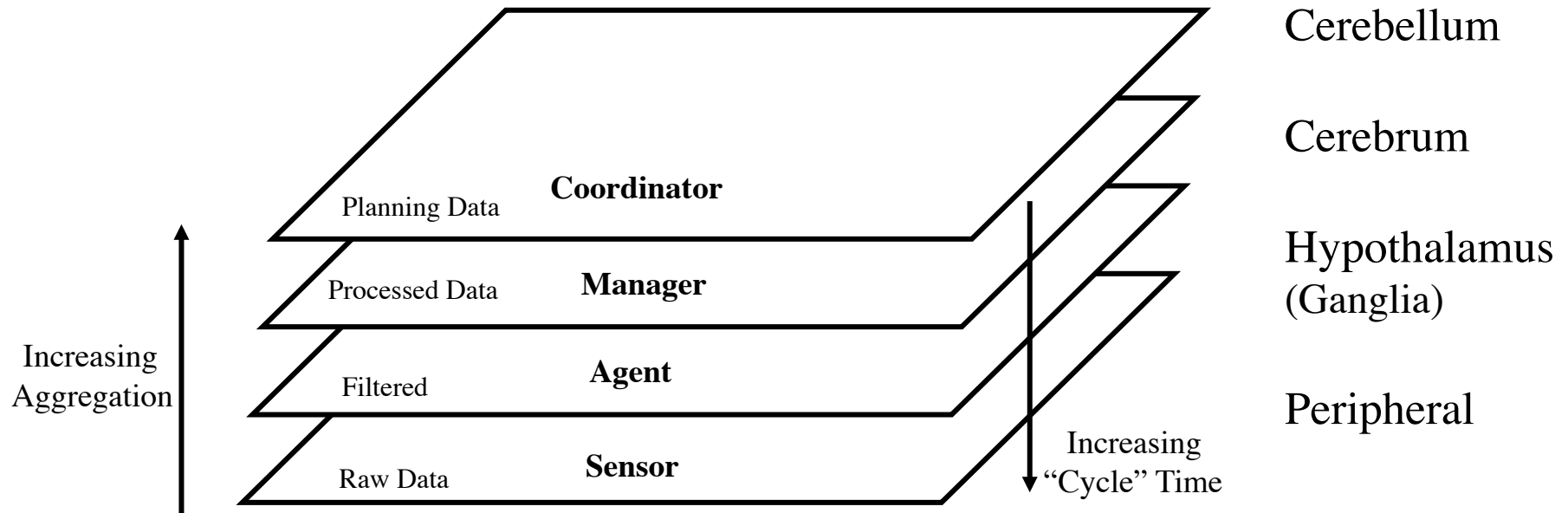
- Application-DMSs will generally be needed for large complex applications and of course, are very application specific, so there is not much we can say beyond the general model.
- A traditional OS is a heterogeneous DAF that includes the peripherals.
 - The traditional device drivers are members of the DAF.
 - In the case of the disk, it might have several members: one, looks like a file system, one that looks like a database, and one that yields track and sector access.
 - And a short step from this to this:

Even More Distributed



- A traditional OS is a heterogeneous DAF that includes the peripherals.
 - Where ever they are.
- Somehow this is much different once you look at the picture.
- An OS is distributed resource manager that in previous years operated under severe connectivity constraints.
- The differences between OS and Network Management becomes a matter of degree.

We Know More About This



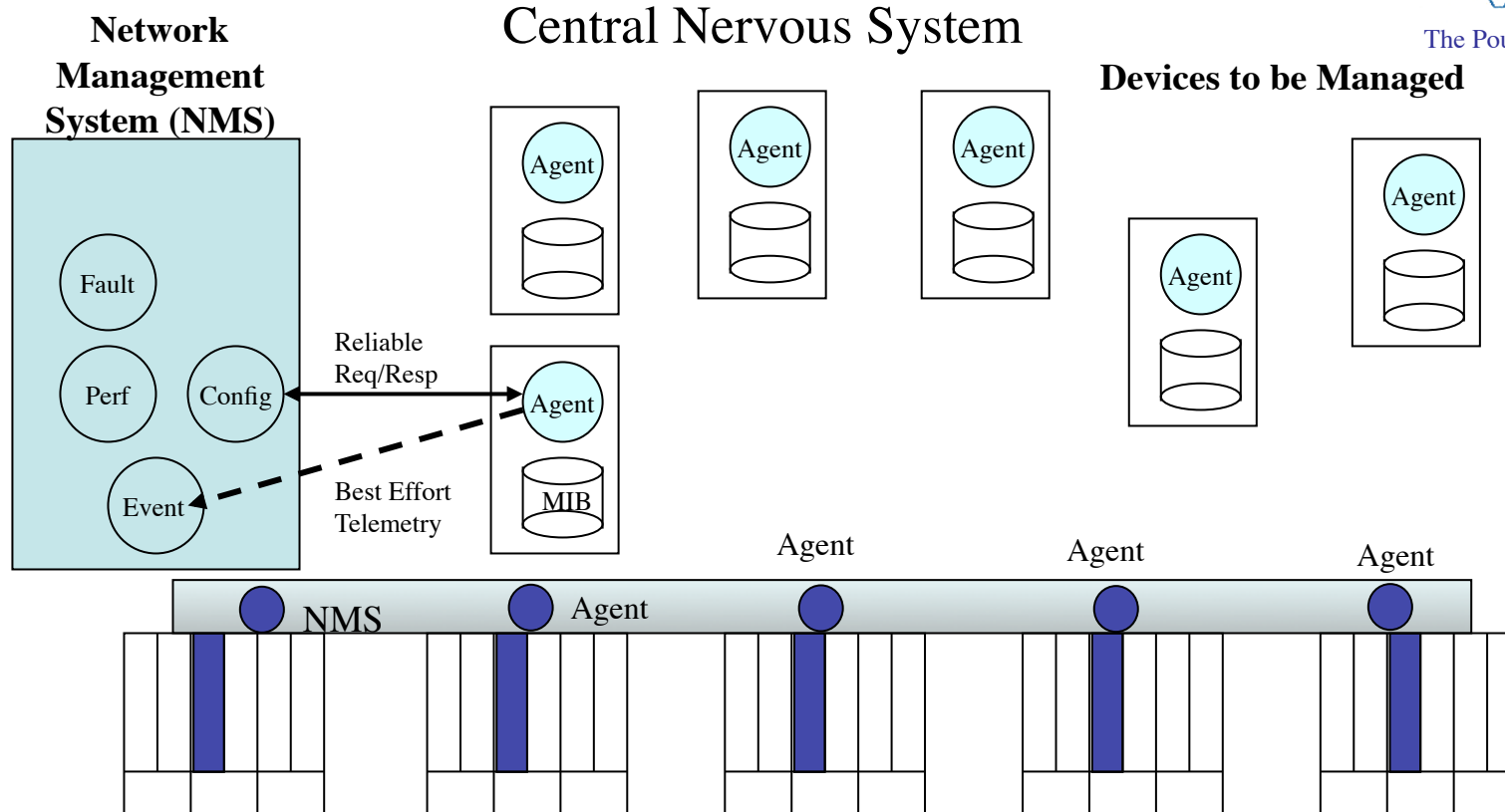
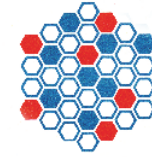
- And down the side were the labels
- This became the core of our approach to Network Management

But More Importantly It was Clear that
Network Management is

Monitor and Repair But not Control

- The whole point is that events are happening too fast for humans to be in the loop. They can manage, but not control.
- Control must be autonomic.

The Management Architecture

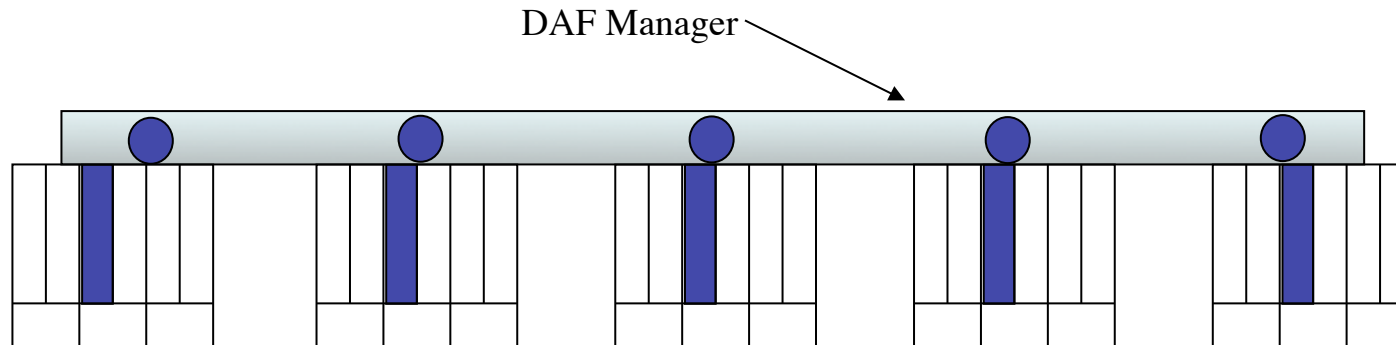


- While there might be managers for distinct subnets (domains) and the subnets might be a hierarchy, the managers were peers.
 - Many talked about managers of managers but there is really nothing for second level managers to do. (that generalizes?)
- Fault Management isn't an app, it is a management system with a small domain.
- Then Realized what was missing: Where's the Homunculus?

Autonomic or Layer Management

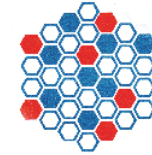
- Clearly Routing was the primary example. It was clear that routing and resource allocation were confused.
 - But there seemed to be so much variation in what the layers did
 - Also resource limitations prevented much practical exploration.
- There are those who believe autonomic is all that is needed:
- This is true, it can be.
 - As long as the complexity never gets beyond that of
 - Mycetozoa, porifera, or coelenterata.
 - slim molds, sponges and maybe jelly fish.
 - It can find local optimal points, but tends to miss global ones.
- But just as in nature, there are interesting configurations along the line from fully distributed to very centralized.

The DAF Management Model Is Perfect for Exploring It



- Have already seen the traditional centralized configuration.
- Could also have configurations where the functionality of the DAPs was more or less the same, OR
- Where some DAPs served as “area coordinators” or ganglia as they are called providing local centralization.
- This is an area for much further exploration.

The Most Important Property for Management

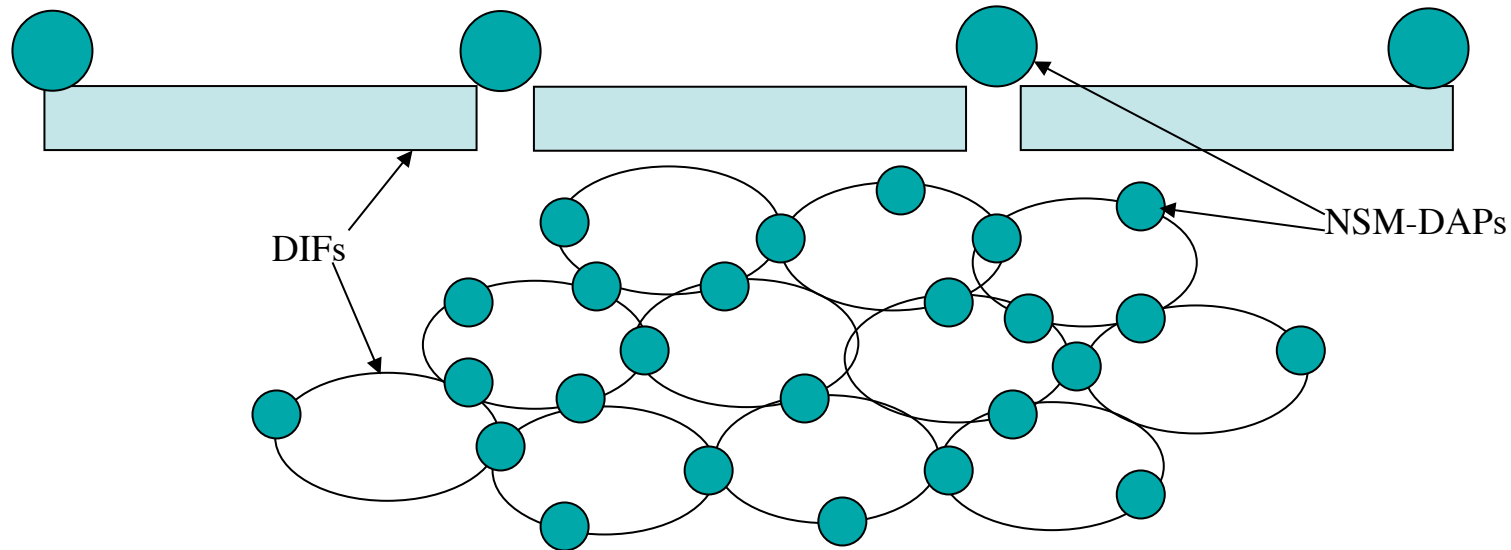


- Commonality, Commonality, Commonality.
 - Reduce the Parts Count.
- Not Necessarily just make everything look alike, but
- Effectively separating the like from the unlike
 - Maximizing invariance and minimizing discontinuities
- Bounding the range of variation (divide and conquer)
- This is what the principles we have uncovered do, and have been
- Embodied in RINA.
 - RINA was not designed to do this. We worked out the principles and then did what they said. (There wasn't that much leeway.)
 - We aren't done. We have pushed commonality into major parts of the model but there are more principles, invariances to find.
 - It is subtle, greatest generality with least constraint, often requires shift in POV

Name Space Management DSMs



Name Space Management (NSM)

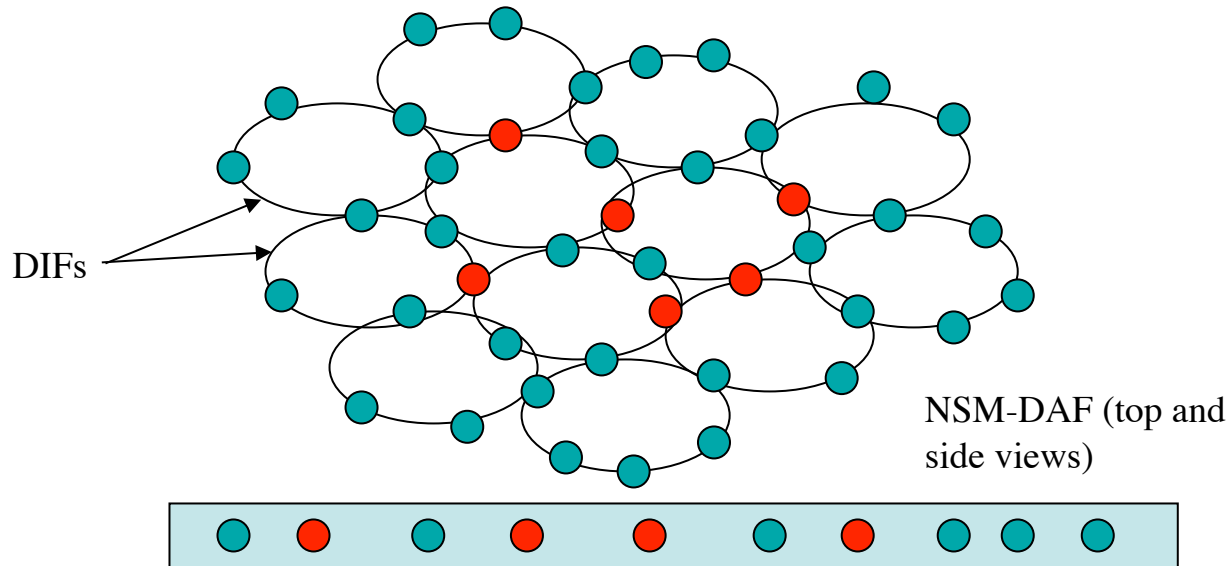


- The IPC Model posits a function that allows the Application Name Space to have a greater scope than any one DIF.
 - Which we have called the Inter-DIF Directory (for lack of a better term)
 - Entity associated with the IPC Management in DAPs may query what applications are available in a system.
 - This forms a graph where the nodes are NSM-DAPs and the arcs are DIFs

NSM-DSMs

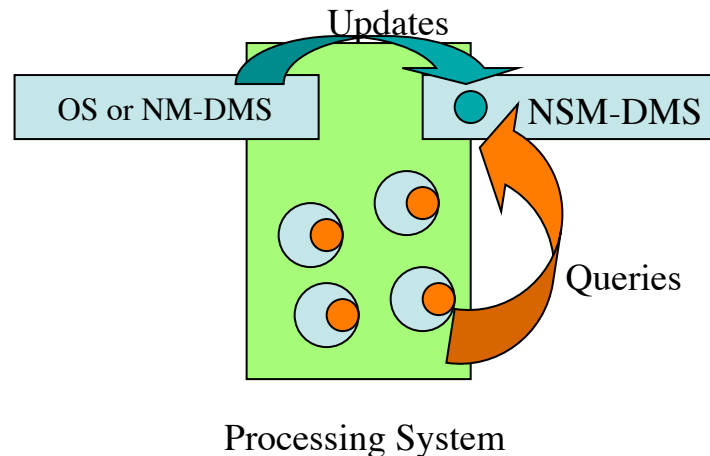
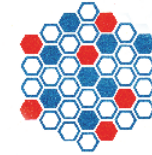
- Considering this a Name Space Management DMS reveals the functions:
 - Authenticate applications that are allowed to query the NSM-DMS
 - Authenticate and authorize entities that are allowed to update or modify the NSM-DMS.
 - Implement the policies for updating and replicating data to meet load and reliability requirements, including creating forwarding tables.
 - Check credentials of a request to determine requestor has access to the requested DAF and if so, return a list of DIFs and supporting DIFs.
 - Manage the name space, determine who gets assigned what.
 - Manage the creation of a common DIF between the requesting and requested DAPs.

NSM-DSMs



- For an environment of any size, we can expect that information on available applications will be organized to shorten search time.
 - Hence some NSM-DAPs will contain only local information: ●
 - While others will be repositories for aggregate information: ●
 - The repositories might be organized by a hierarchy, DHTs, the Dewey Decimal System, etc.
 - This implies two kinds of forwarding tables:
 - Find the next repository, either aggregate or local. ●
 - Forward among NSM-DAPs to get to those repositories. ● ●

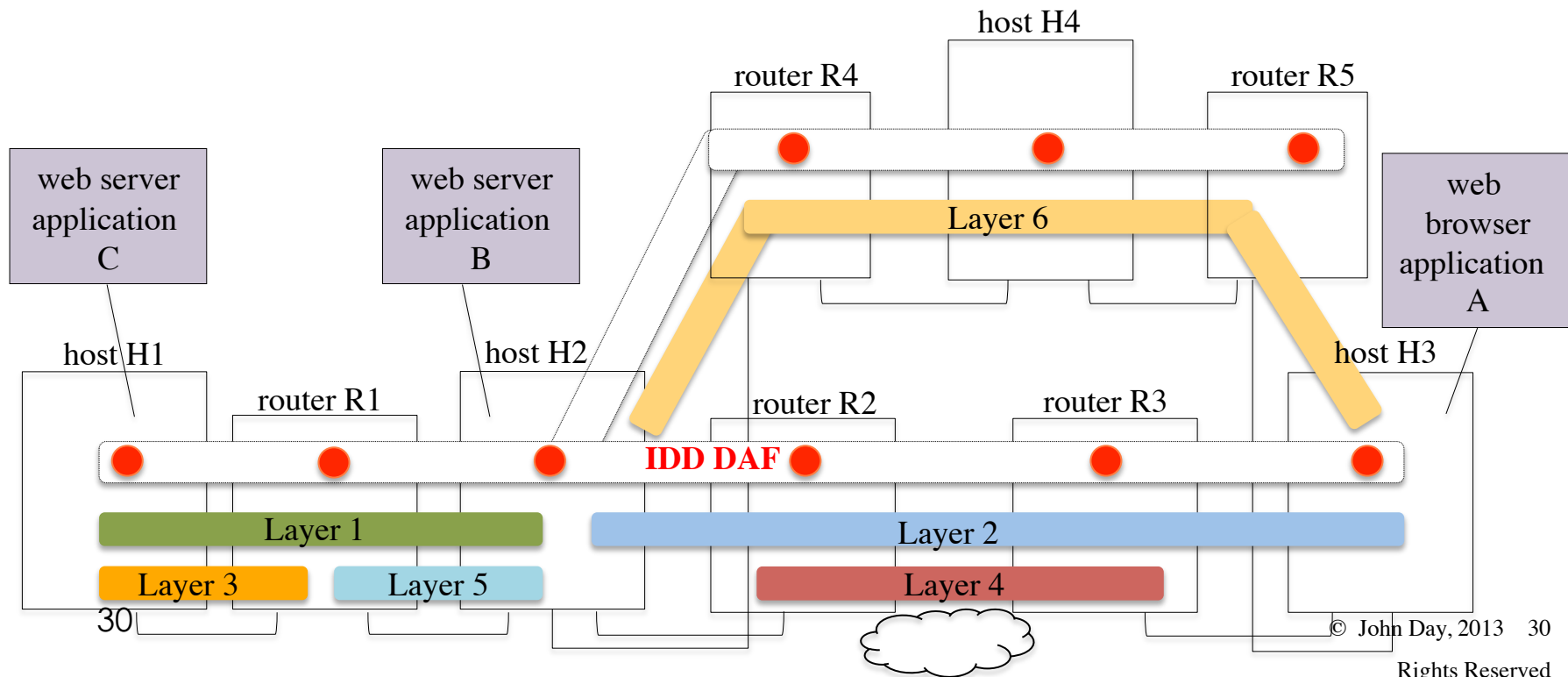
NSM-DMSs



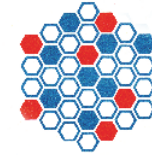
- Clearly there is a potential scaling problem here, if we are not careful.
- For large systems, a management system (either an OS-DMS or NM-DMS) will be responsible for access control domains.
 - These DMSs will be authorized to update or modify information aggregated with a NSM-DMS, will provide the local NSM-DAP, and participate in creating or joining new DIFs.
 - Everything else will be a NSM-client only, i.e. can only submit queries.
 - May not be considered a member of the NSM-DAF or a lesser member.
- For small systems, it degenerates into the DAF structure.

Discovery of the application

- ▶ Forwarding of the request between the peer **IDDs** until the destination application is found or the pre-defined termination condition is met



IDD Information



- Naming / synonyms
- Neighbor Table
- Search Table
- Directory

Naming Information

IDD Application Process Name
synonyms (optional)

Search Table

Application Process Name	List of Peer IDDs Application Process Names
--------------------------	---

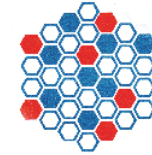
Neighbor Table

Peer IDD Application Process Name	List of Peers IDDs Application Process Names
-----------------------------------	--

Directory

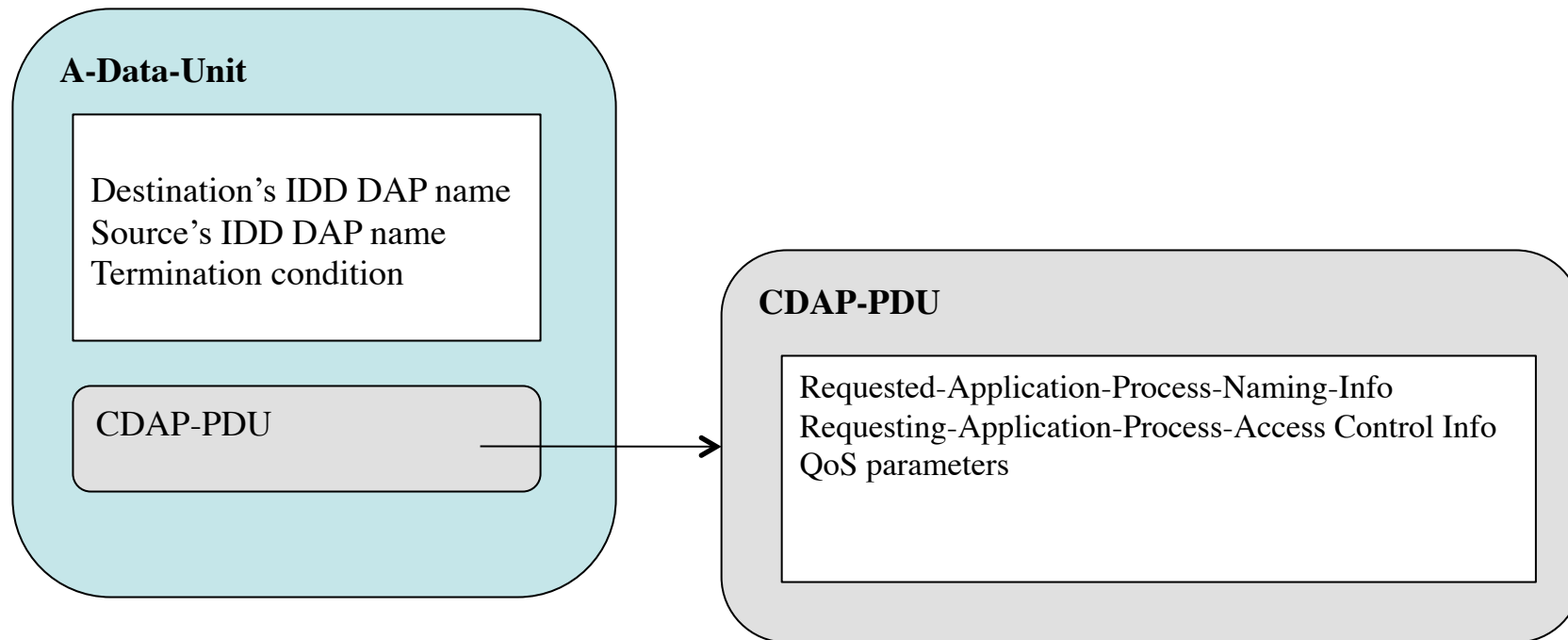
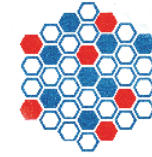
Application Process { Name, Access Control Information }
List of supporting DIFs { Name, Access Control Information, supported QoS }

What the IDD request looks like?

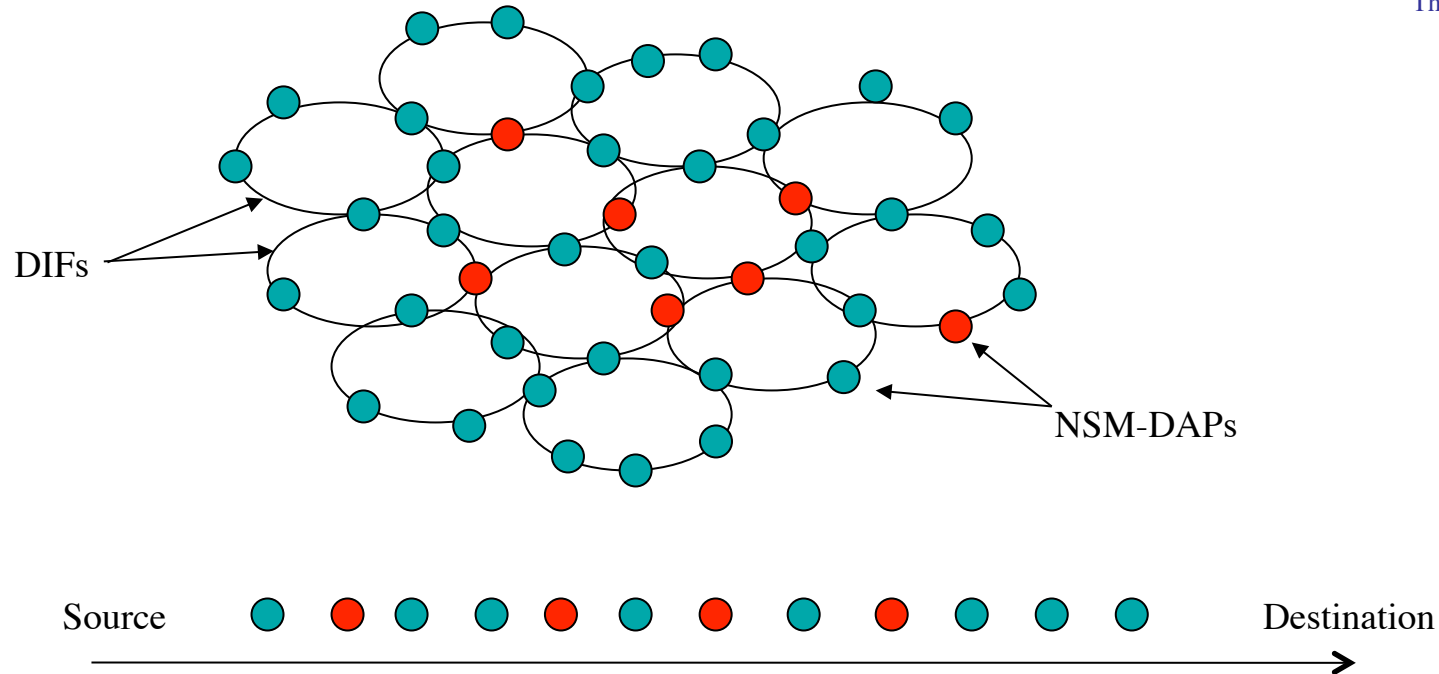


- A CDAP Read Request for an IDD-Record
 - IDD-Request**
 - requested-Application-Process-Naming-Information
 - requesting-Application-Process-Access Control Information,
 - QoS parameters
- The CDAP Read Request can be encapsulated in an A-Unit-Data
 - A-Unit-Data**
 - destination's IDD DAP name
 - source's IDD DAP name
 - termination condition (e.g. hop count)
 - CDAP-PDU

How is it forwarded?



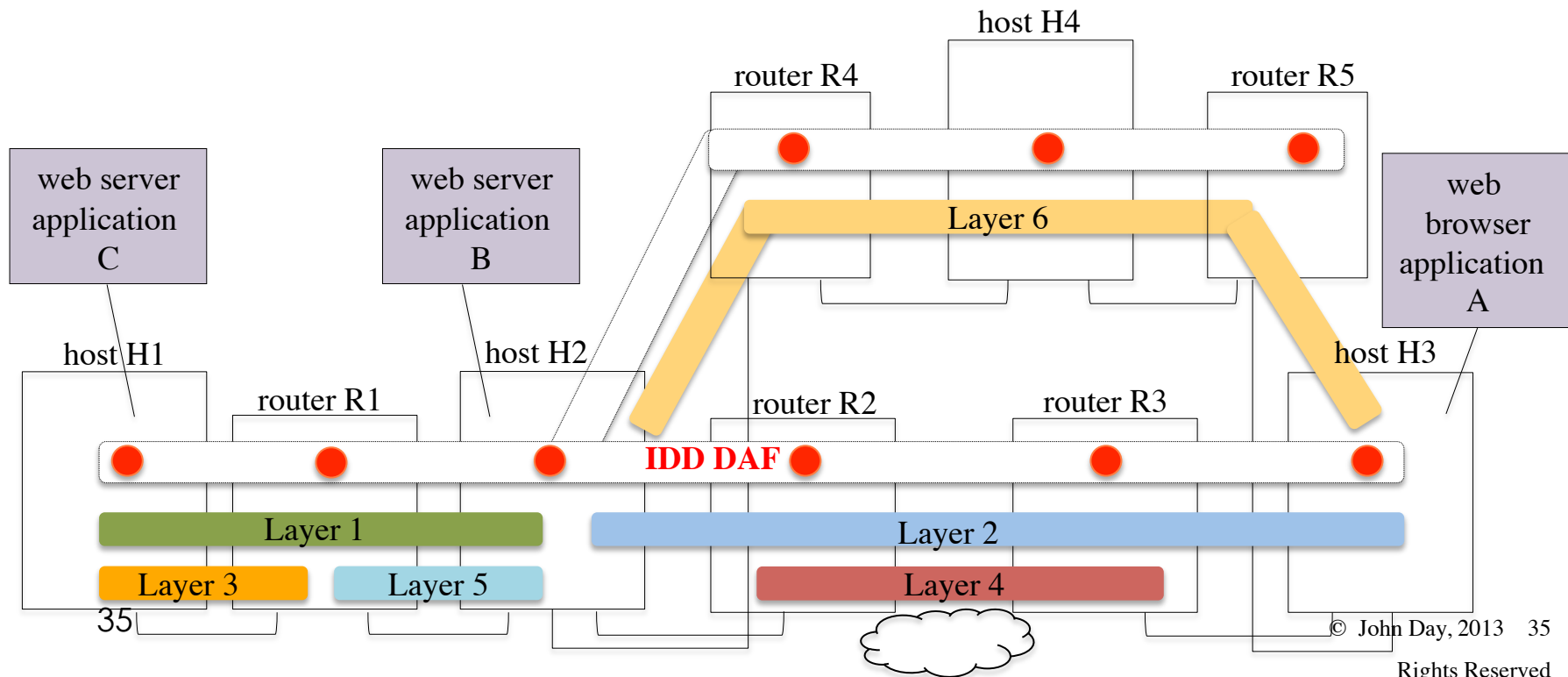
How is it forwarded?



- From any DAP to the other you forward A-Data-Units
- In the first, the last and all the red DAPs you process the CDAP PDU
- Only in the destination IDD DAP (last one) you do a CDAP Read for an IDD-Record

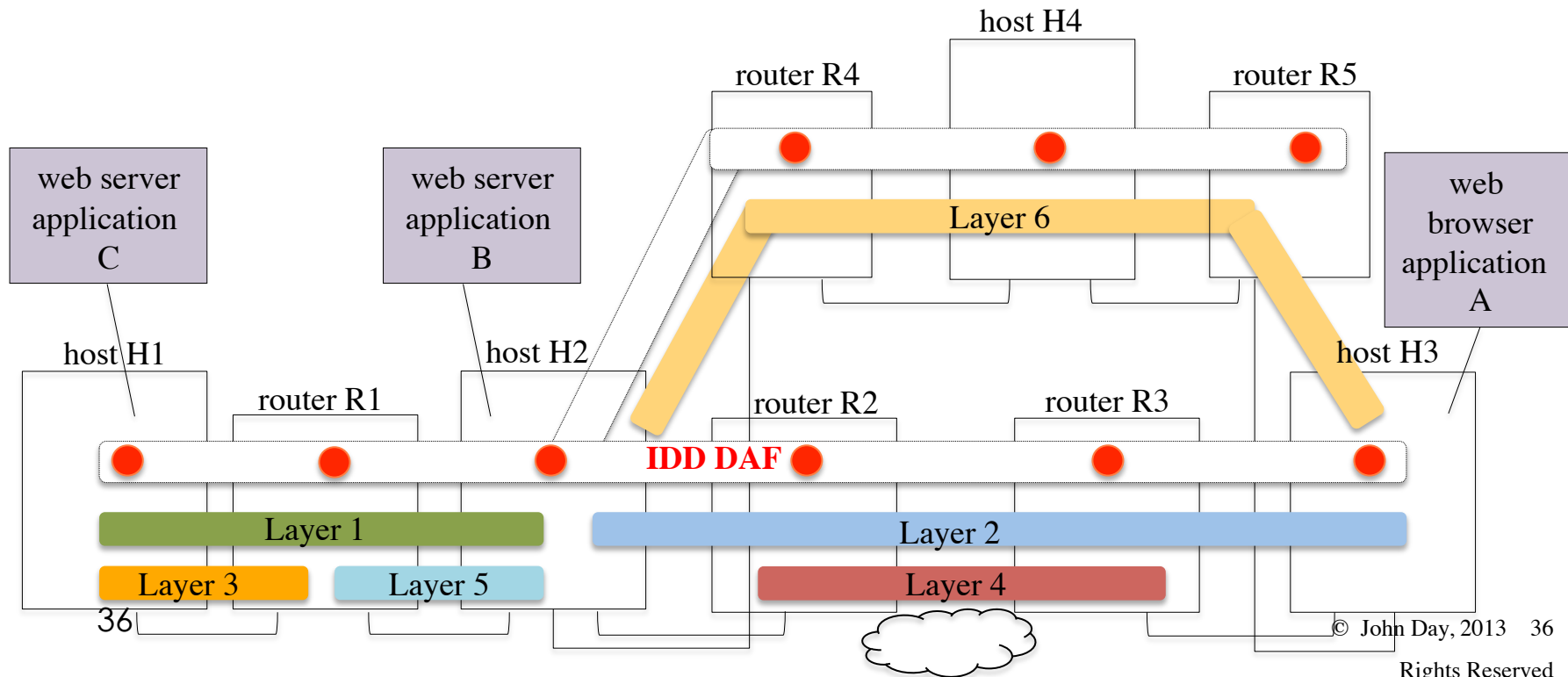
Discovery of the application

- Confirmation that the requested application is available in the destination system and authorization check that the requesting application has the rights to access it



Creation of the supporting DIF

- ▶ A DIF supporting the communication between the two user applications has to be found
- ▶ This either involves creating a new DIF from scratch or expanding (joining) an existing one so that it spans from the source to the destination system



Implications

- There is no application discovery mechanism in the Internet today, not just pointers to where to search next as today with DNS
- Applications do not have to be in the same layer to discover each other, especially not on the same one layer as with IP
- Elimination of the need for layers with large address spaces
- No need for a single application namespace. Name spaces can be tailored to environments.
- Greater security by having multiple application namespaces and by better compartmentalization without impairing reachability

Another Interesting Pattern

- Notice that the pattern exhibited by the NSM-DSM of:
 - Look up among distributed data bases (NSM-repositories) followed by the creation of distributed shared state (DIF).
- Has precisely the same structure as the Flow Allocator:
 - Look up among distributed data bases (Directory) followed by the creation of distributed shared state (Connection).
- Which has precisely the same structure as Routing:
 - Look up (computation) among distributed data bases (forwarding table) followed by the creation of distributed shared state (routes).
- The first involves with multiple management domains and DIFs
- The second involves possibly multiple management domains and one DIF
- While the third is one management domain and one DIF.
- There may be another collapse here.

Questions?