Clark's "Funnel" Reconsidered

Suppose we wanted to design a shared platform, like FABRIC, that supported storage, networking and computational services. In order to achieve application portability we would design it with a spanning layer, meaning a common interface that all applications would use to access its most fundamental resources. Then we would be faced with the task of designing that spanning layer.

If we were guided by the Deployment Scalability Tradeoff [Beck, "On the Hourglass Model," CACM, July 2019], we would choose a set of necessary applications (perhaps defined by a set of application characteristics that we decide to support). We would then design our spanning layer to be as weak, simple, general and resource limited as possible while supporting this set of applications. The reasons for including these four characteristics are different. The weaker the spanning layer the greater the number of possible underlying services that can support it. The simpler and more general the spanning layer the greater the number of possible applications that can be implemented using it. Finally, the tighter the limits we place on the size of a single allocation of resources, the less stringent we need to be about making each allocation (that is, we can rely more on dynamic contention and avoid per-use billing).

Since our supposed spanning layer includes services as seemingly disparate as storage, networking and computation, a natural approach is to simply bundle the interfaces to those respources as currently used by applications into a single combined interface, introducing overlay logic to coordinate their combined use. In such a design applications would access specific interfaces for a file, database or other mass storage service (e.g., NFS); a network transfer service (e.g., TCP/IP); and a computation service (e.g., the x86 ISA). The spanning layer could be constructed by making specific choices among available products or by constructing generic overlay services implemented using a variety of products.

The problem with this approach is that the resulting spanning layer may not be weak, simple, generic and/or resource limited. Application interfaces tend to be at the top of complex vertical silos of services, aggregating fine-grained underlying resources into highly specialized and optimized interfaces. In cases where distributed solutions must be constructed using elements from different silos, accessing that functionality through high level application interfaces is often awkward or inefficient.

We can recognize the effects of such overlay convergence in the history of converged infrastructure, including previous research testbeds. Active Networking applied storage and computation by invoking substantial operations on streams delivered using TCP/IP. Web caches and middleboxes acted at a higher level, storing and computing on HTTP responses consisting of collections of typed digital objects. Delay Tolerant Networking builds on reliable communication links and mass storage services which store and forward bundles of files. Content Delivery Networks and Internet Content/Named Data Networking use storage of collections of typed digital objects and overlay multicast together with sophisticated metadata management, policy and subscription services. PlanetLab was based on the Linux kernel interface (itself a bundling of storage, network and computation) and secure file transfer services augmented by sophisticated account and slice management. Software Defined Networking as used in GENI is only a policy routing augmentation of the IP service --- converged resources were made available through GENI Racks, which were a bundling of application level services. The spanning layer of the Computational Grid was another such bundling of application service interfaces. None of these examples were highly successful in achieving deployment scalability of the kind exhibited by the Internet and the Unix kernel interface within their respective application communities.

How can we possibly design an interface that can be used to implement the disparate functions of storage, networking and computation without simply bundling the familiar and effective interfaces that applications now use in an unwieldy overlay? The other available strategy is to build a common underlay which provides the primitive services that are used to implement all of them, and to use them to build the

functions of current silos and of other high level services that span those silos. If there is sufficient commonality in those fundamental services, then there would be a hope of defining a weak, simple, general and at least optionally resource limited spanning layer at that lower level. The potential stumbling blocks are twofold: finding this primitive basis, and building systems out of it that perform well enough.

The abstraction that we propose is the primitive unit of storage/buffering/memory, which we refer to generically as a buffer. In storage it may be long-lived, stable and densely implemented (as in a block); in networking it may be fast and small (as in a line buffer); in computation it may be fast and close to functional units (as in a memory frame). In current architectures and systems we segregate these different types of buffers according to the functions they perform, optimizing their use within each silo and perhaps completely disallow access across silos. We propose the buffer as the fundamental building block and the basis of interoperability.

The idea of defining a network spanning layer at a level below IP was suggested in [Clark, "Interoperation, Open Interfaces and Protocol Architectures," in The Unpredictable Certainty: White Papers, 1997], although the candidate lower layer being considered was specifically ATM. "The virtue of this approach is that a single technology solution might be able to offer a sophisticated range of services and thus support an even wider range of applications than, for example, the Internet approach. So the funnel, while narrow at the bottom might be wide at the top." [see Figure 1(C)] Clark went on to say that "An approach that spans a broad range of networks is needed because there never has been, and in practice never will be, a single network technology sufficiently powerful and general to meet everyone's needs. The ATM solution, if fully embraced, would require replacing all the legacy networks such as Ethernet, token ring, the telephone digital hierarchy, and so on."

What Clark seems not to have considered feasible in 1997 was the possibility of creating an approach that spans a broad range of network technologies at a level low enough to allow the construction of a sophisticated range of services and thus support an even wider range of services than, for example, the Internet. In other words, he did not consider lowering the spanning layer below ATM, and modeling radically primitive services sufficiently powerful and general to meet everyone's needs. He did not foresee a future in which token ring and digital telephony were subsumed by the Ethernet model, and where heterogeneity could be enhanced by settling on a virtualized interface to local node services.

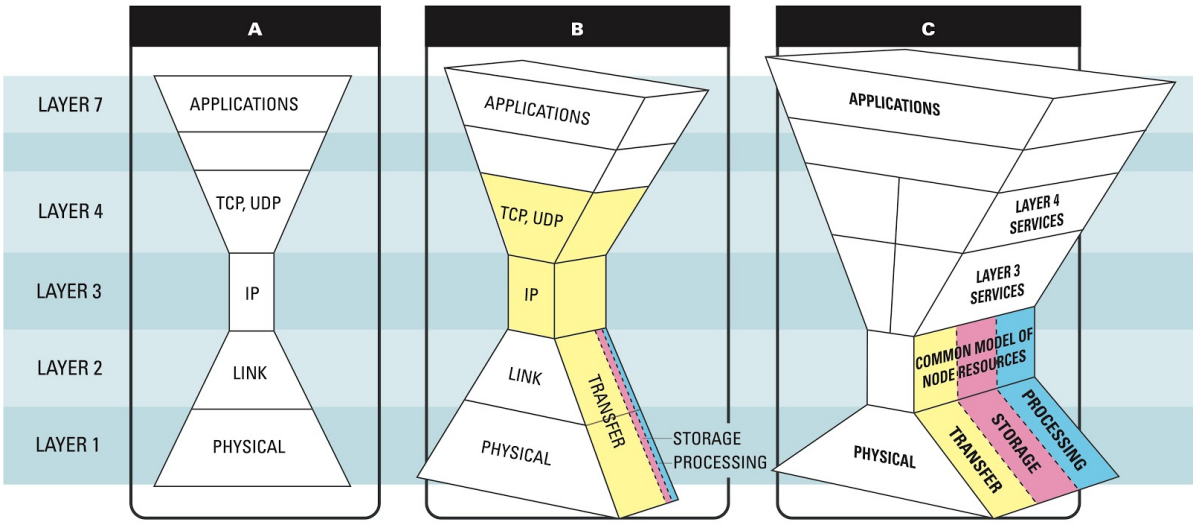

Figure 1: Internet hourglass and a virtualized interpretation of "Clark's Funnel" [Beck et al., "Interoperable Convergence of Storage, Networking and Computation", FICC 2019]