Pagemarks used to keep track of reading progress

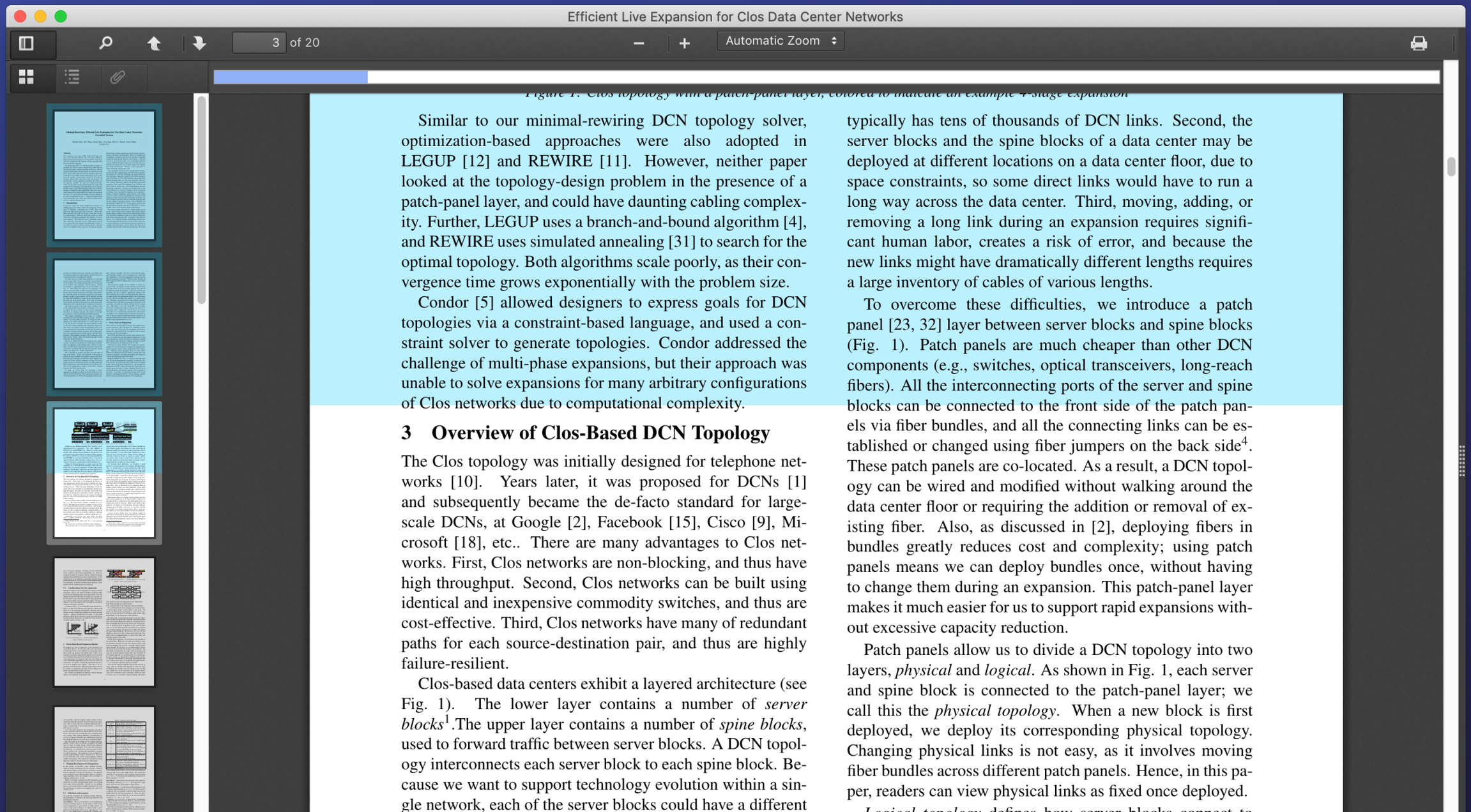

Figure 1: Clos topology with a patch-panel layer, colored to indicate an example 4-stage expansion

Similar to our minimal-rewiring DCN topology solver, optimization-based approaches were also adopted in LEGUP [12] and REWIRE [11]. However, neither paper looked at the topology-design problem in the presence of a patch-panel layer, and could have daunting cabling complexity. Further, LEGUP uses a branch-and-bound algorithm [4], and REWIRE uses simulated annealing [31] to search for the optimal topology. Both algorithms scale poorly, as their convergence time grows exponentially with the problem size.

Condor [5] allowed designers to express goals for DCN topologies via a constraint-based language, and used a constraint solver to generate topologies. Condor addressed the challenge of multi-phase expansions, but their approach was unable to solve expansions for many arbitrary configurations of Clos networks due to computational complexity.

3 Overview of Clos-Based DCN Topology

The Clos topology was initially designed for telephone networks [10]. Years later, it was proposed for DCNs [1] and subsequently became the de-facto standard for large-scale DCNs, at Google [2], Facebook [15], Cisco [9], Microsoft [18], etc.. There are many advantages to Clos networks. First, Clos networks are non-blocking, and thus have high throughput. Second, Clos networks can be built using identical and inexpensive commodity switches, and thus are cost-effective. Third, Clos networks have many of redundant paths for each source-destination pair, and thus are highly failure-resilient.

Clos-based data centers exhibit a layered architecture (see Fig. 1). The lower layer contains a number of *server blocks*[1].The upper layer contains a number of *spine blocks*, used to forward traffic between server blocks. A DCN topology interconnects each server block to each spine block. Because we want to support technology evolution within a single network, each of the server blocks could have a different typically has tens of thousands of DCN links. Second, the server blocks and the spine blocks of a data center may be deployed at different locations on a data center floor, due to space constraints, so some direct links would have to run a long way across the data center. Third, moving, adding, or removing a long link during an expansion requires significant human labor, creates a risk of error, and because the new links might have dramatically different lengths requires a large inventory of cables of various lengths.

To overcome these difficulties, we introduce a patch panel [23, 32] layer between server blocks and spine blocks (Fig. 1). Patch panels are much cheaper than other DCN components (e.g., switches, optical transceivers, long-reach fibers). All the interconnecting ports of the server and spine blocks can be connected to the front side of the patch panels via fiber bundles, and all the connecting links can be established or changed using fiber jumpers on the back side[4]. These patch panels are co-located. As a result, a DCN topology can be wired and modified without walking around the data center floor or requiring the addition or removal of existing fiber. Also, as discussed in [2], deploying fibers in bundles greatly reduces cost and complexity; using patch panels means we can deploy bundles once, without having to change them during an expansion. This patch-panel layer makes it much easier for us to support rapid expansions without excessive capacity reduction.

Patch panels allow us to divide a DCN topology into two layers, *physical* and *logical*. As shown in Fig. 1, each server and spine block is connected to the patch-panel layer; we call this the *physical topology*. When a new block is first deployed, we deploy its corresponding physical topology. Changing physical links is not easy, as it involves moving fiber bundles across different patch panels. Hence, in this paper, readers can view physical links as fixed once deployed.

Logical topology defines how server blocks connect to