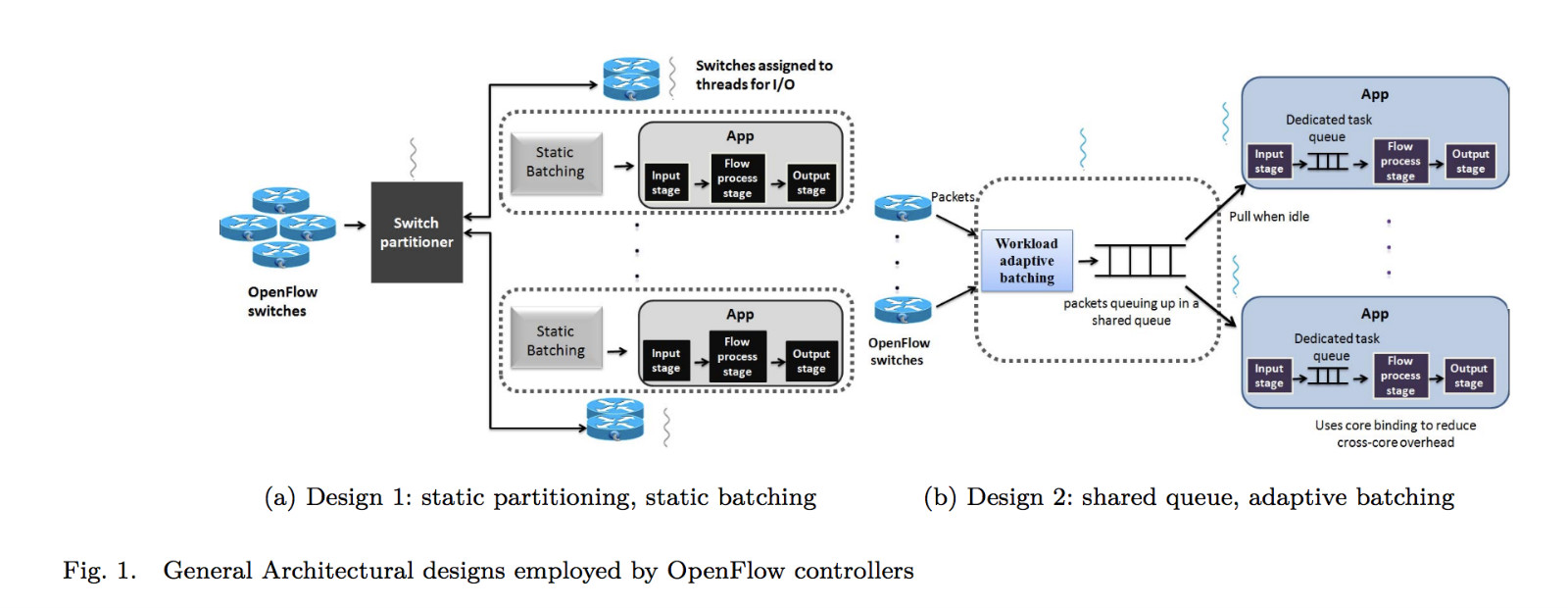
Week 20

How different controllers leverage the above designs and the impact of these design choices on the controllers’ performances.



A. Maestro

Maestro is a multi-threaded OpenFlow controller whose architecture is identical to the design shown in Figure 1(b). OpenFlow packets are received from the sockets by the main thread and put in the shared raw-packet queue.2 Packet batching is used but the number of bytes to be read is not static and depends on the present workload (work-load adaptive batching). Maestro is the only publicly-available controller which uses task batching so that worker threads pull a batch of tasks to process multiple flow-requests in a single execution. Output batching technique is used to send packets out in which packets belonging to the same destination are grouped together and sent using a single socket system call.

B. NOX-MT

NOX-MT has a multi-threaded architectural design similar to Figure 1(a). It uses the Boost::Asio [8] libraries for network and low-level I/O programming. As shown in Figure 1(a), Boost::Asio acts as a switch partitioner and is responsible for distributing the connecting OpenFlow switches to worker threads statically. Static packet batching technique is used to reduce frequent read system calls. The incoming packets are processed one-by-one (no task batching) and then batched together in case of high control traffic before they are sent out. Static input batching helps NOX-MT in achieving a high throughput performance but affects its latency.

C. FloodLight

Floodlight follows the architectural design given in Figure 1(a) and uses Java Netty for handling multithreaded

2A recent technical report discusses alternate design techniques instead of using shared-queue design.

sockets. Floodlight uses oversubscription of threads to ensure that the cores are not underutilized. Static switch partitioning technique is used through a fixed size read buffer and by assigning a fixed number of switches to each thread. After per-packet processing, output batching is performed to reduce the overhead of system calls for each individual packet.

D. Beacon

Beacon is a Java-based OpenFlow controller with an architectural design similar to Figure 1(a). If n number of threads are configured to be run, the controller spawns n+1 threads, where the additional thread is responsible for listening to incoming switch connections and partitioning them among the worker threads. Beacon uses a static approach in which a fixed number of switches are assigned to a worker thread. Worker threads use static packet batching to serve the requests from the connected switches. Once the packets are processed and ready to be sent, Beacon in its default mode uses write coalescing and allows only one write per I/O select loop to reduce the overhead of socket system calls for each individual OpenFlow message. Alternatively, an immediate mode can be enabled in which the controller attempts a socket write for every outgoing OpenFlow message waiting to be written to the switch to reduce the per-packet latency. Static partitioning and input batching improve its throughput in the default mode.