Distributed Eratosthenes' Sieve

Runtimes for concurrency and distribution

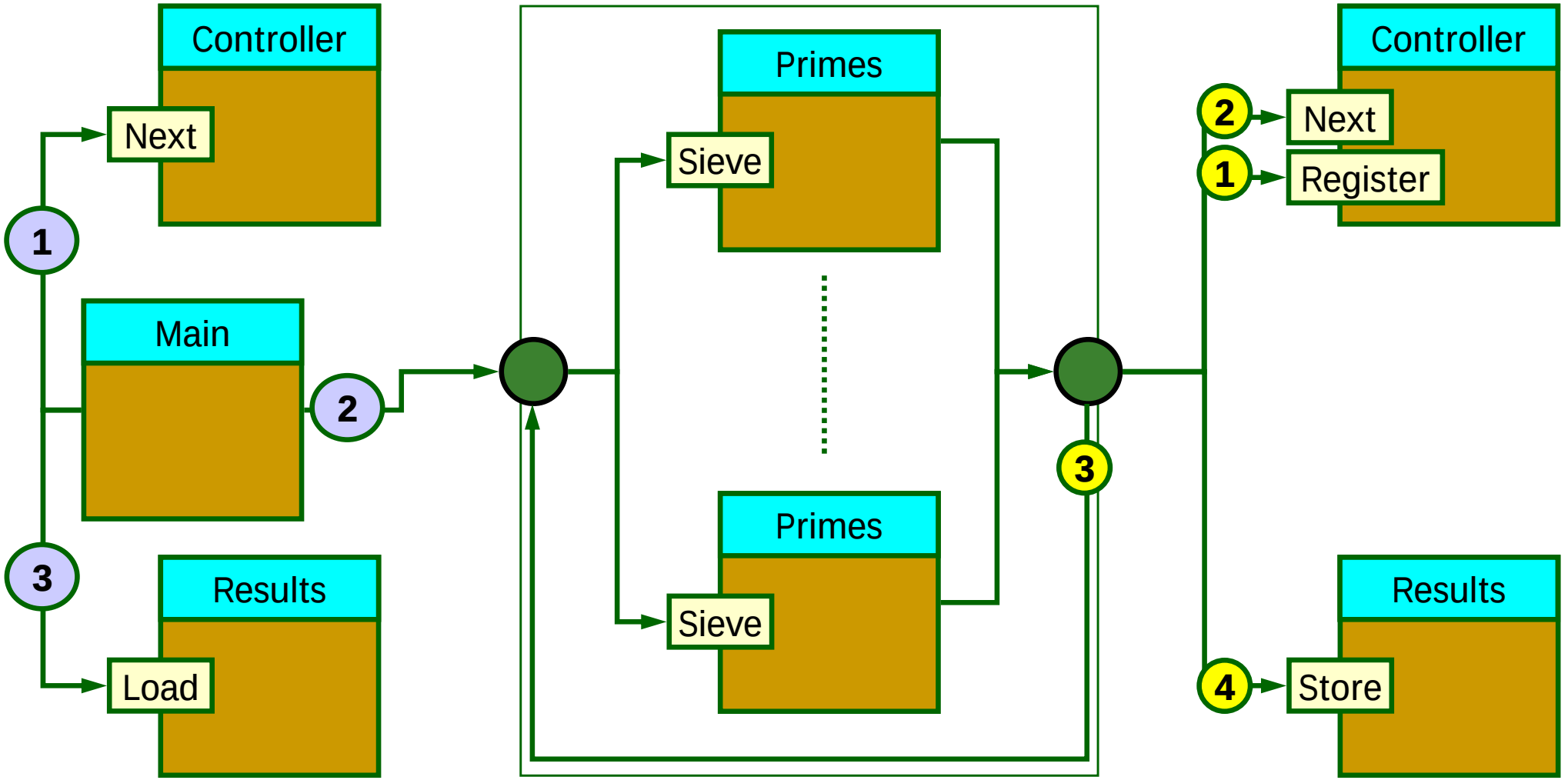
Tullio Vardanega, tullio.vardanega@unipd.it

Academic year 2020/2021

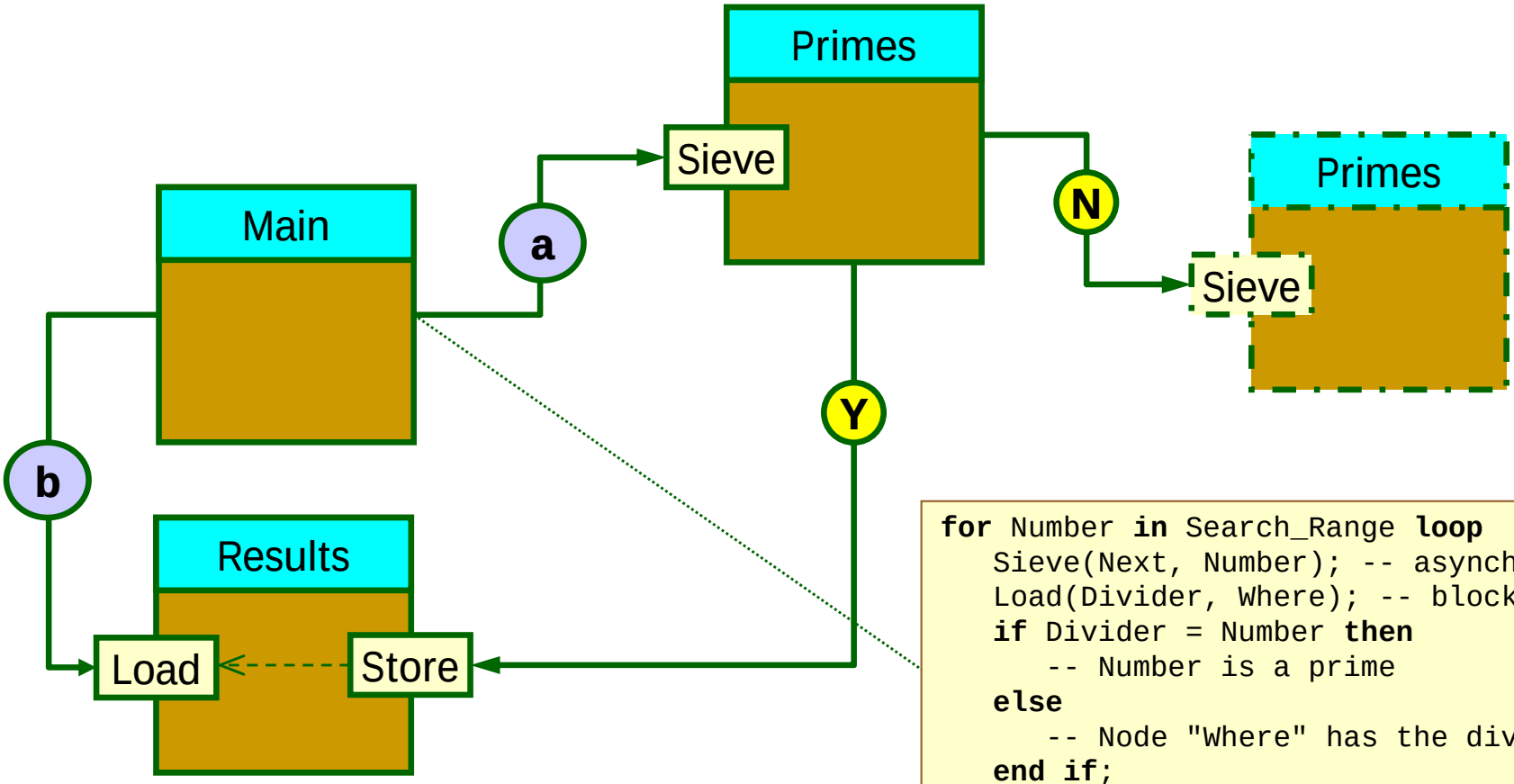
Distributed algorithm – 1

- Using a recursive descent plan in a distributed system is evidently impractical
  - We need a solution that employs a **static pool** of Sieve units and visits them circularly
- We build a **ring** topology as an overlay network
  - Each ring member holds a disjoint subset of prime numbers
- Each new query visits the ring and is checked against each member's subset one value at a time
  - A query that makes full round carries a new prime number
- Main issues query **asynchronously** and then waits for corresponding verdict on **blocking** call

Distributed algorithm – 2

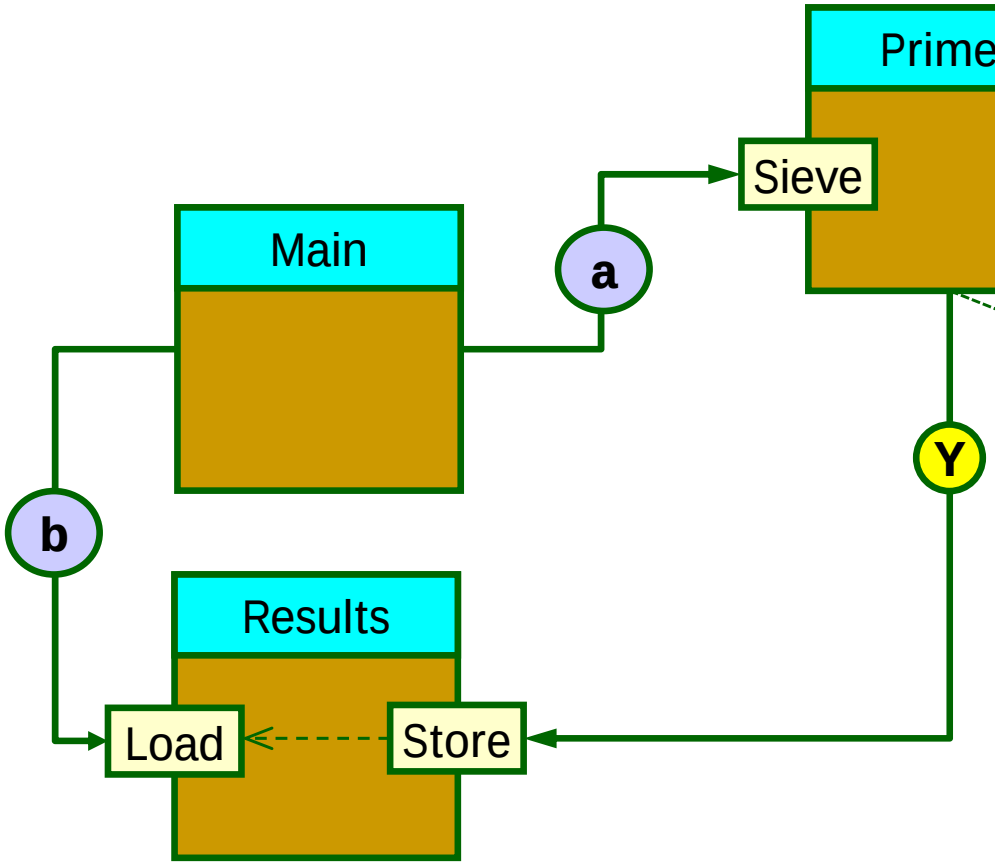


Distributed algorithm – 3




```
for Number in Search_Range loop
   Sieve(Next, Number); -- asynchronous
   Load(Divider, Where); -- blocking
   if Divider = Number then
      -- Number is a prime
   else
      -- Node "Where" has the divisor
   end if;
end loop;
```

Distributed algorithm – 4




```
if Current <= Tail then -- sentinel check
   if (Number mod Prime(Current)) = 0 then
      -- we have a divisor so we record it
      Store(Prime(Current), Self);
   else -- we have no divisor so we ask next
      Current := Current + 1;
      Sieve(Next, Number);
   end if;
else – full round, we have a new prime
   Tail := Tail + 1;
   Prime(Tail) := Number;
   Store(Prime(Tail), Self);
end if;
```

Distributed algorithm – 5

- The bags of primes held by individual ring members are disjoint
  - Their contents are **not** consecutive
- New primes get added to the bag where the highest prime is tried last
  - This implies that each ring member may be queried multiple times
- Hence, the query call issued by any ring member is potentially indirectly **recursive**
  - Callee uses a sentinel value to tell recursive call apart
  - When sentinel value is updated is critical to that effect

System architecture