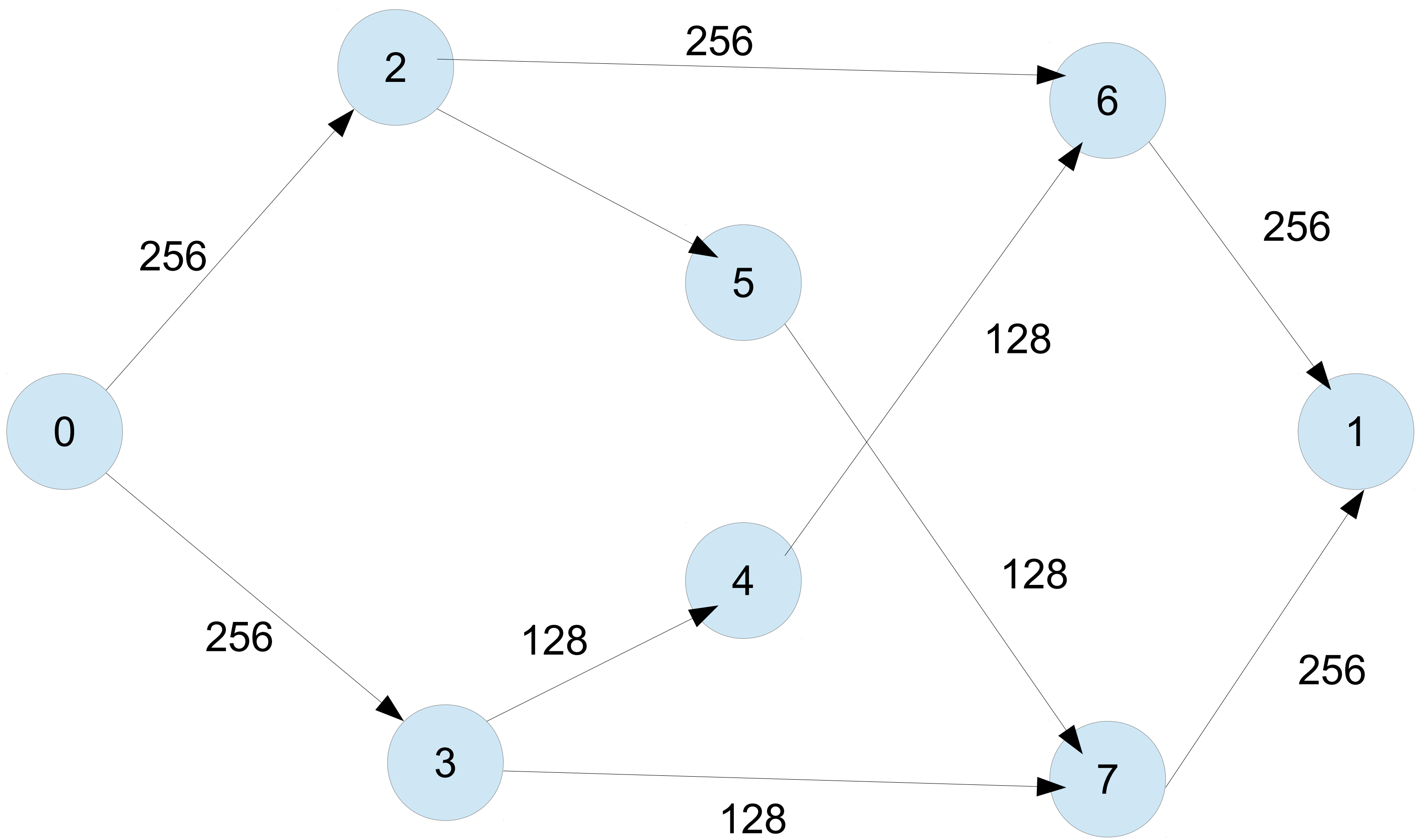
MPM (Malhotra, Pramod Kumar, Maheshwari) algorithm for finding the maximum flow

- **STEP 1:** setup auxiliary network
- **STEP 2:** find the vertex n with minimum throughput h
- **STEP 3:** Push h units of flow from n through shortest paths n, $n_{i+1, \ldots}$ t, if vertex is not the sink
- **STEP 4:** Pull h units of flow from source to n
- * Vertices that have throughput 0 together with the edges connect to them should be removed.

Algorithm

```
while True: # stage
    construct auxiliary graph AN(f) from residual N(f)
    if t not reachable from s:
        return maximum flow
    while True: # constructing a blocking flow
        (1) delete all nodes with zero throughput in AN(f)
            together with the incident arcs, and update the
            throughput of the neighboring nodes
        (2) if s or t not in AN(f): break
        (3) v = node in AN(f) with minimum throughput h
        (4) push h units of flow from v to t
        (5) pull h units of flow from s to v
```

Original Network



Flow is 0 in all edges

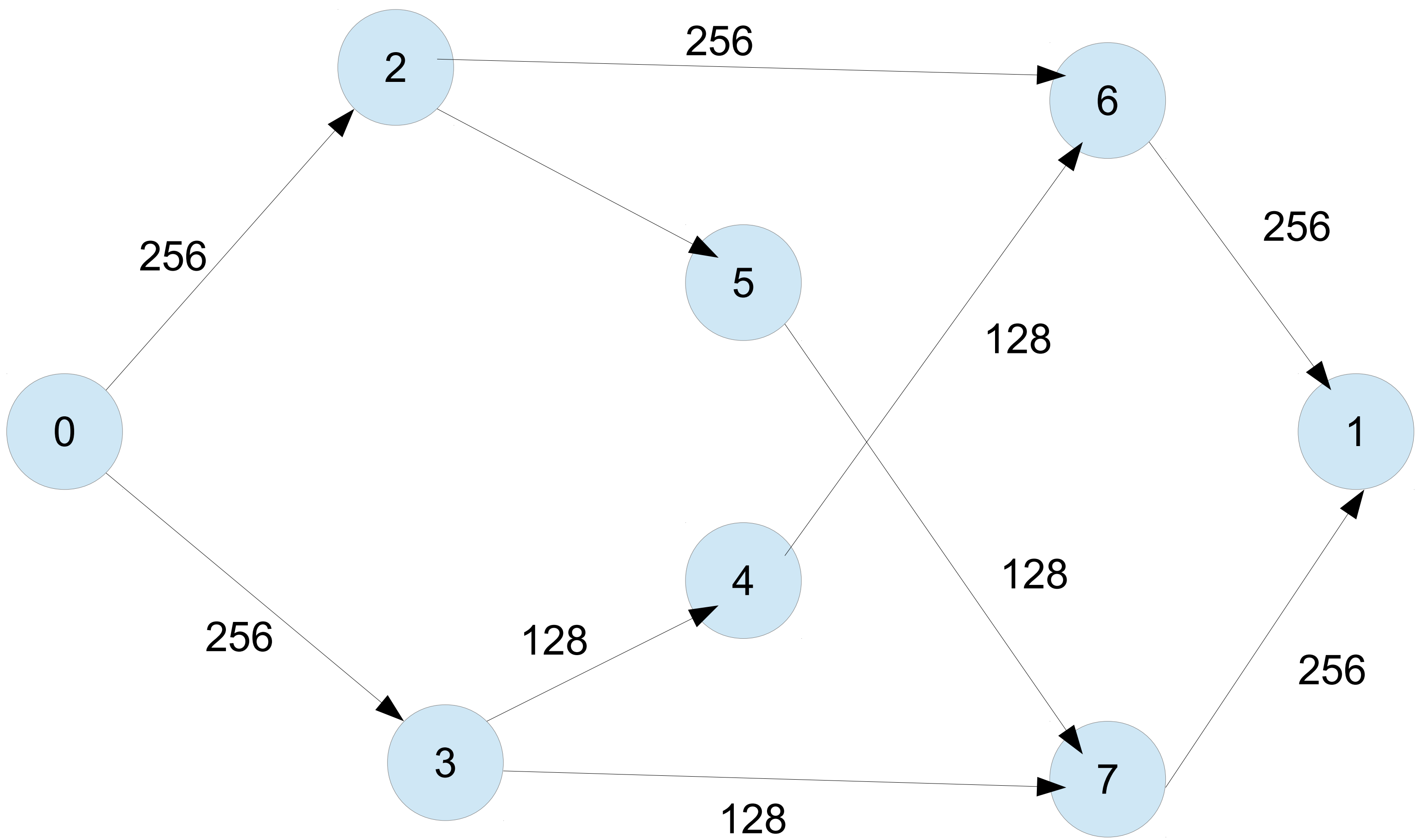
Building Residual Graph

- Residual graph $N(f)$ consists of the following arcs:

  a) $(u, v) \in A(f)$ with $ac(u, v) = b(u, v) - f(u, v)$
  if $(u, v) \in A$ and $b(u, v) - f(u, v) > 0$

  b) $(v, u) \in A(f)$ with $ac(v, u) = f(u, v)$
  if $(u, v) \in A$ and $f(u, v) > 0$

Residual Network

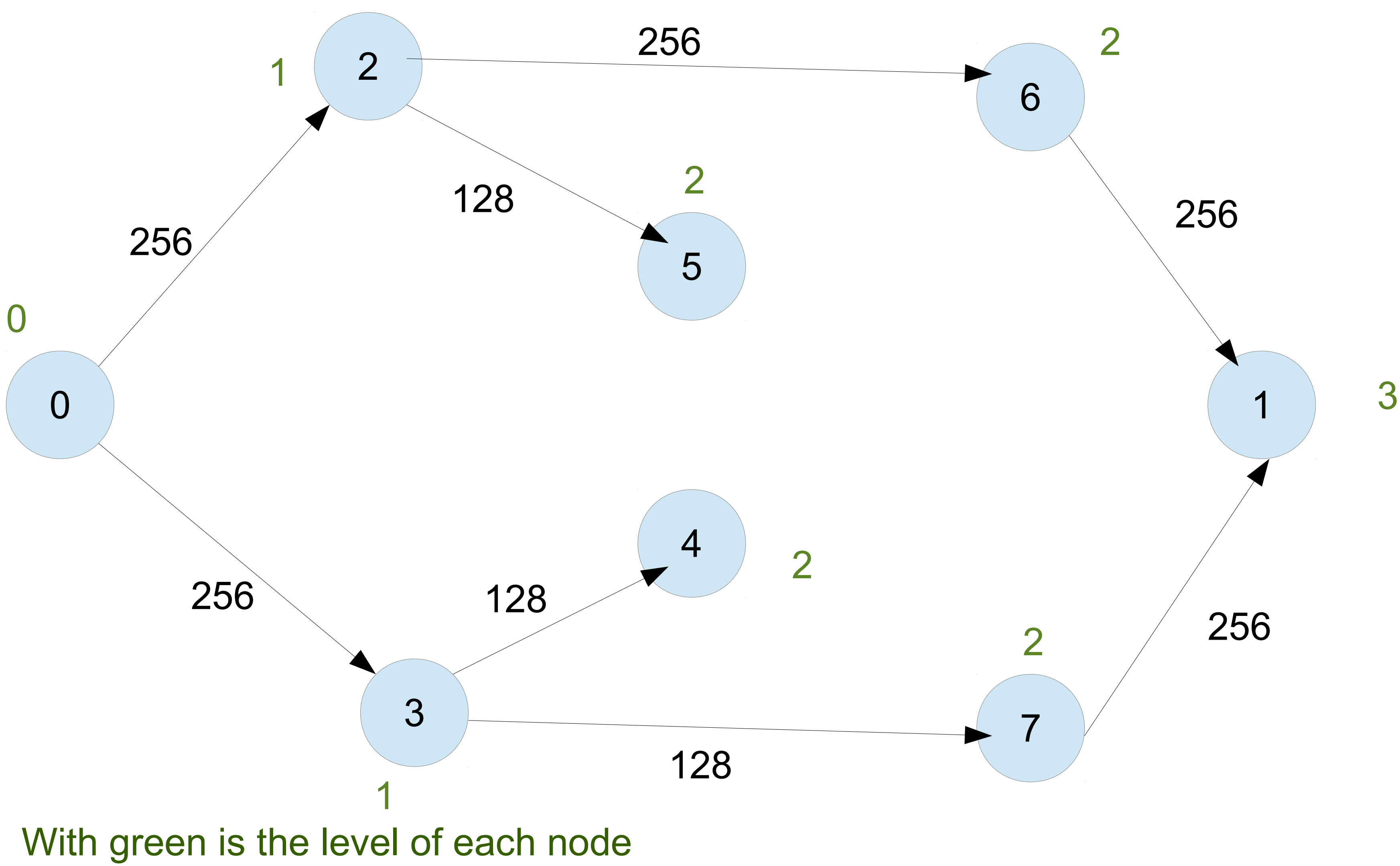


Flow is 0 in all edges

Building Auxiliary Network

- We create it while carrying out the breadth first search on the residual network by keeping only the arcs that leads us to new nodes and only the nodes that are at lower level than t.

Auxiliary Network



Finding a blocking flow

- If the flow value |g| can not be improved in AN without introducing new back or same level arcs, then g is call a blocking(maximal) flow with respect to the current flow f.
- The main part of the algorithm is to find a blocking flow in the auxiliary network AN

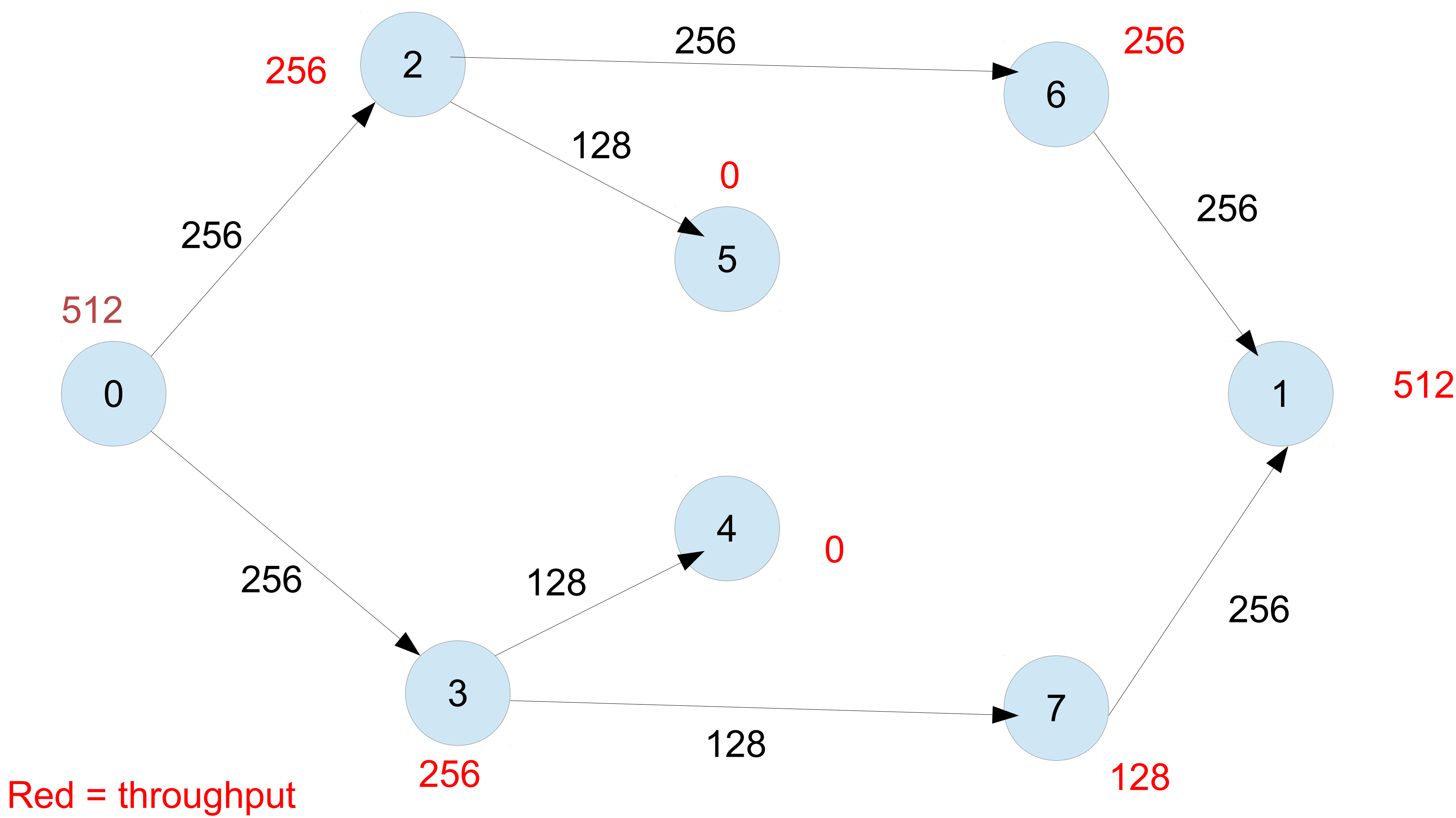
Finding a blocking Flow

- Throughput of a node u is the maximum amount of flow that can be pushed through u.
- If u != sink and u != source:

  throughput[u] = min(incomingCapacity,outgoingCapacity)

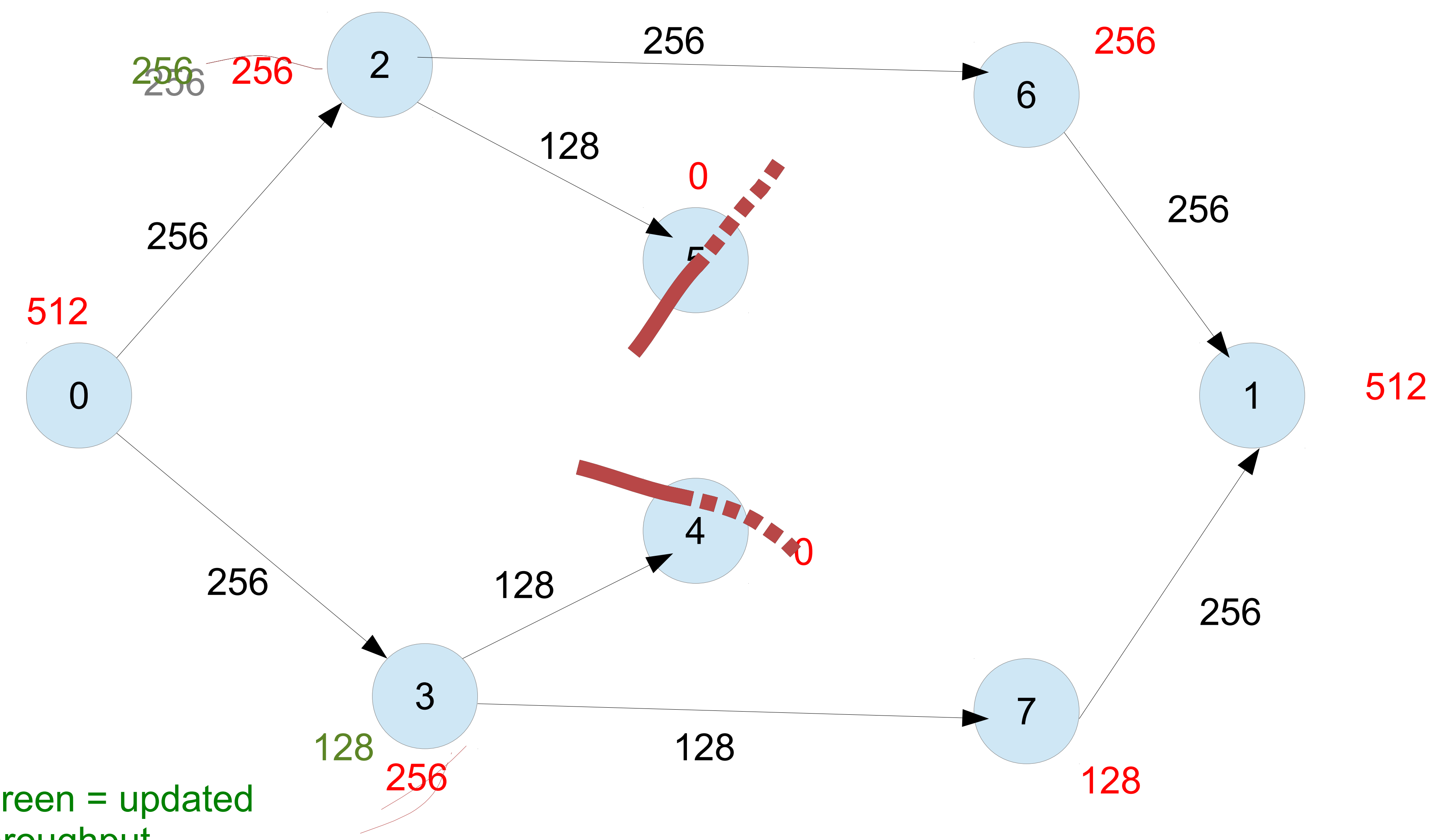
  - If u == source: throughput[u] = outgoingCapacity
  - If u == sink: throughput[u] = incomingCapacity

Throughput



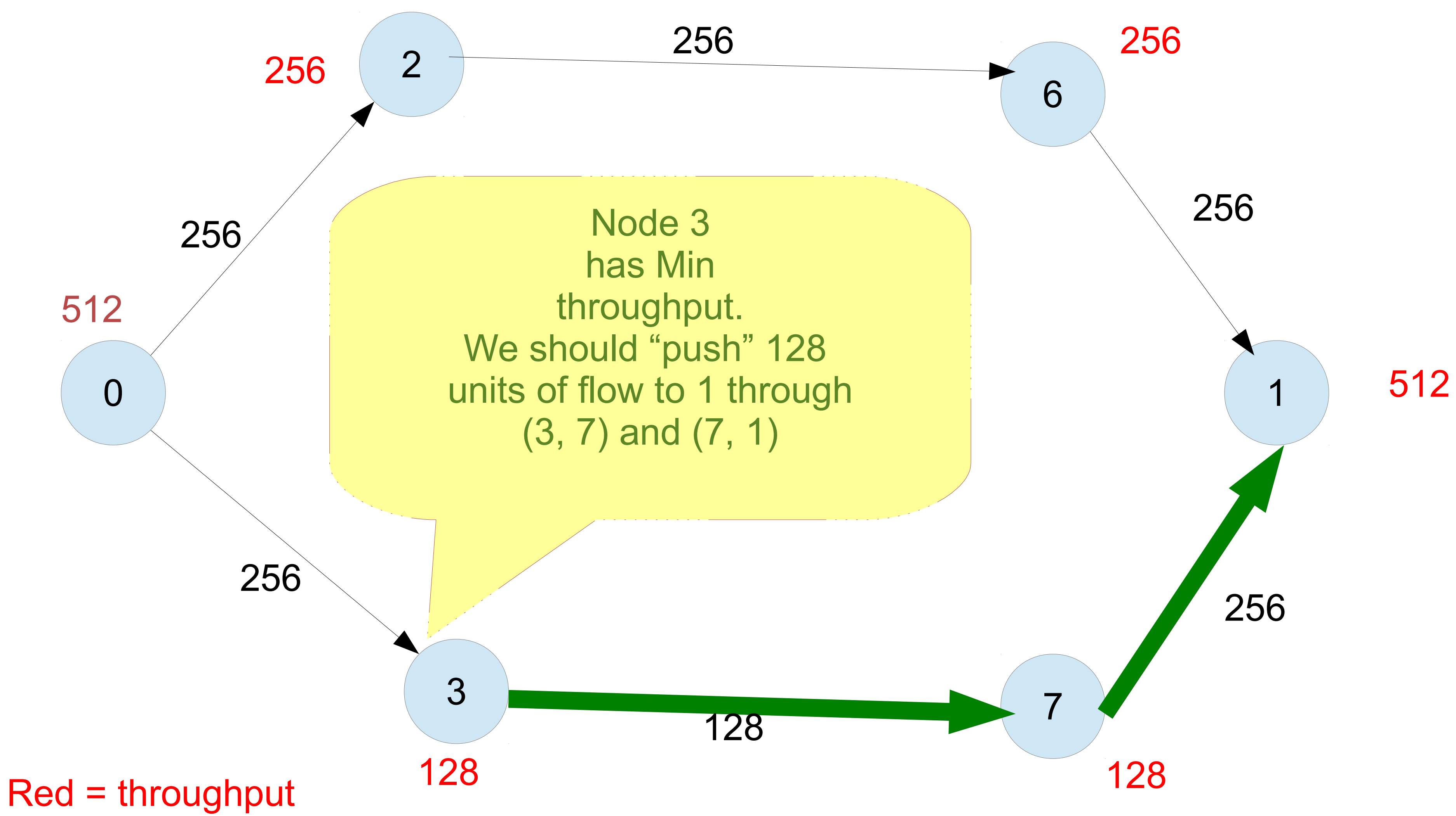
0
1
2
3
4
5
6
7
256
128
256
256
256
128
128
256
512
256
0
256
512
0
256
128
Red = throughput

Deleting nodes with 0 throughput

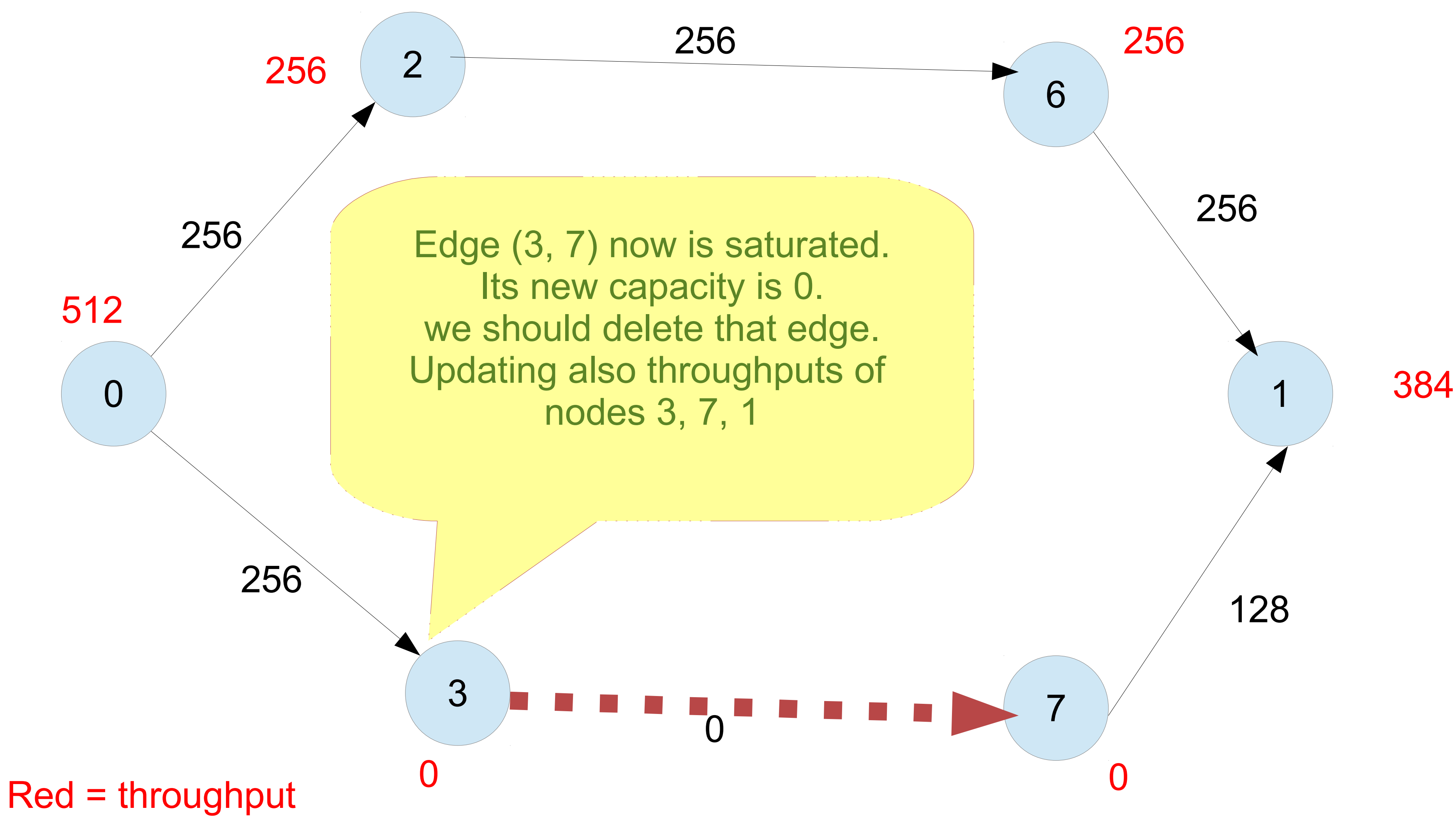


Green = updated throughput

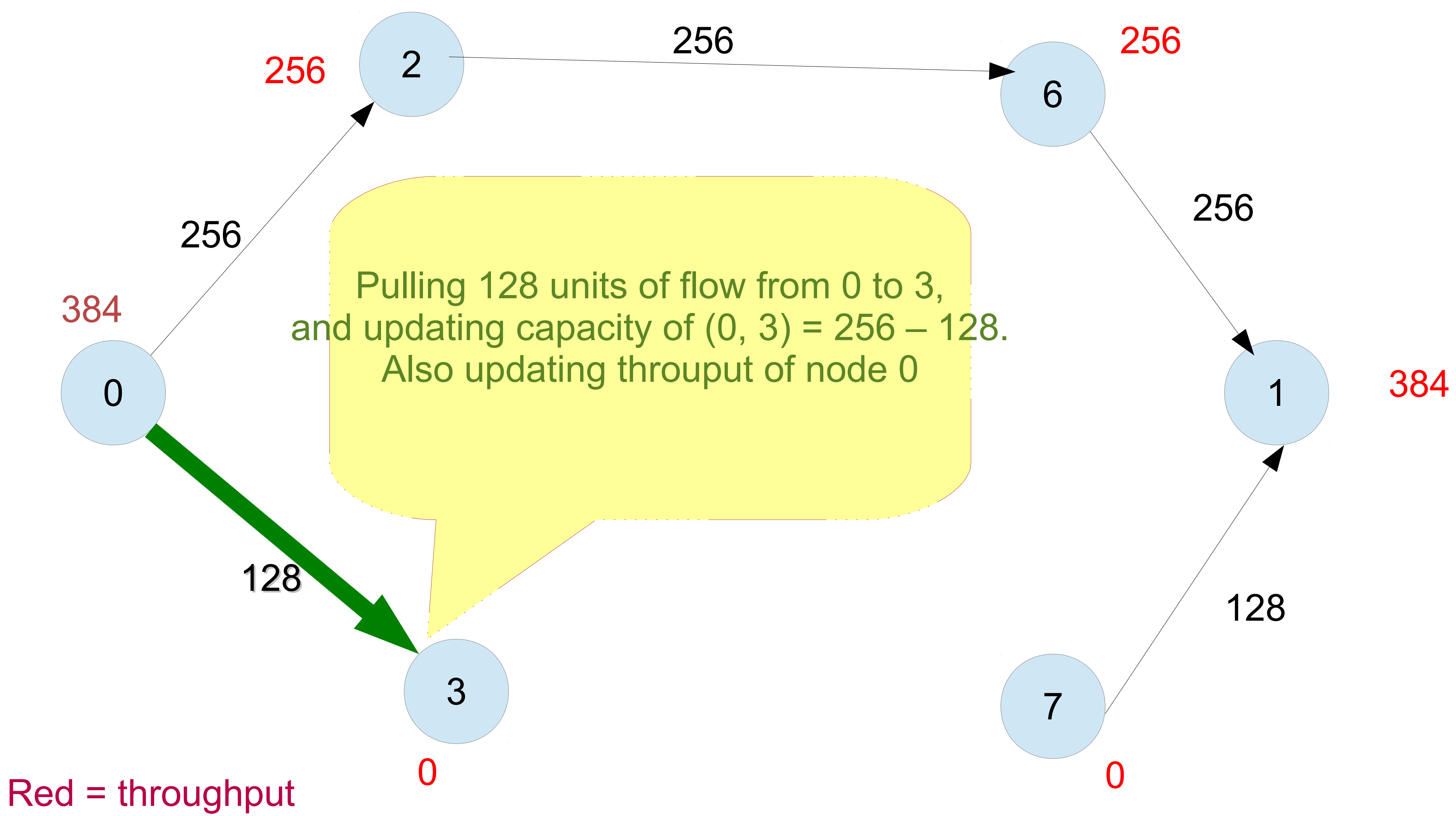
Pushing from node with min throughput



Pushing from node with min throughput (2)



Pulling flow from s to node 3

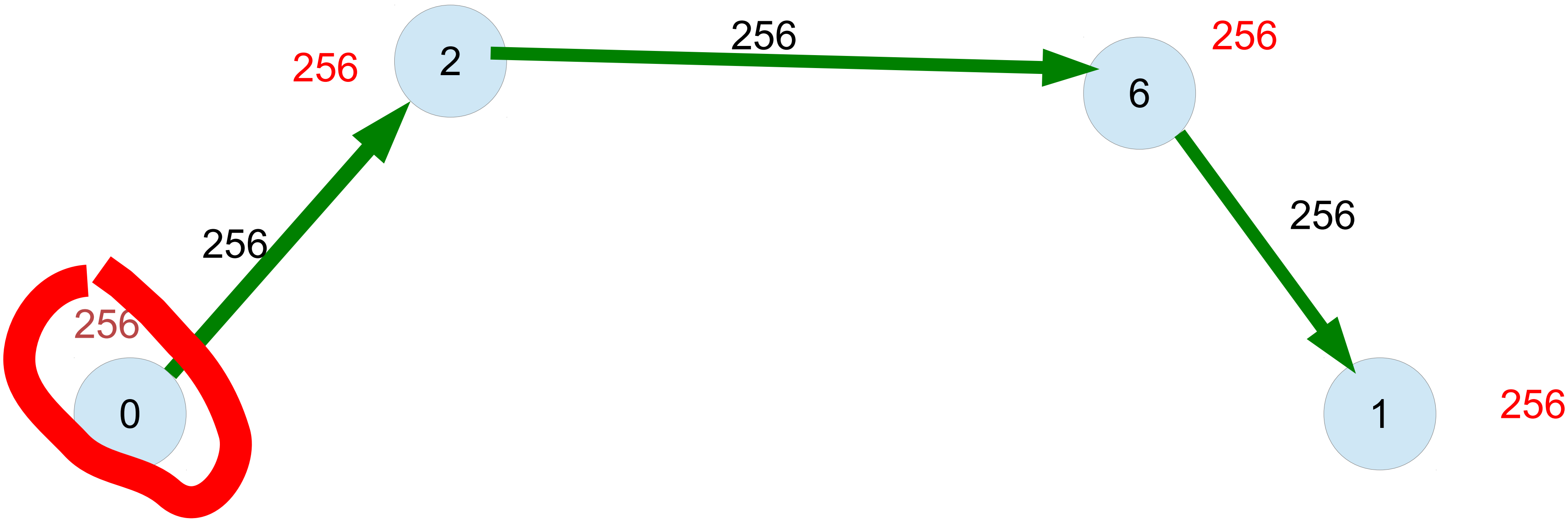


Deleting nodes with zero throughput

Deleted nodes 3 and 7 and updated throughput of nodes 1 and 0

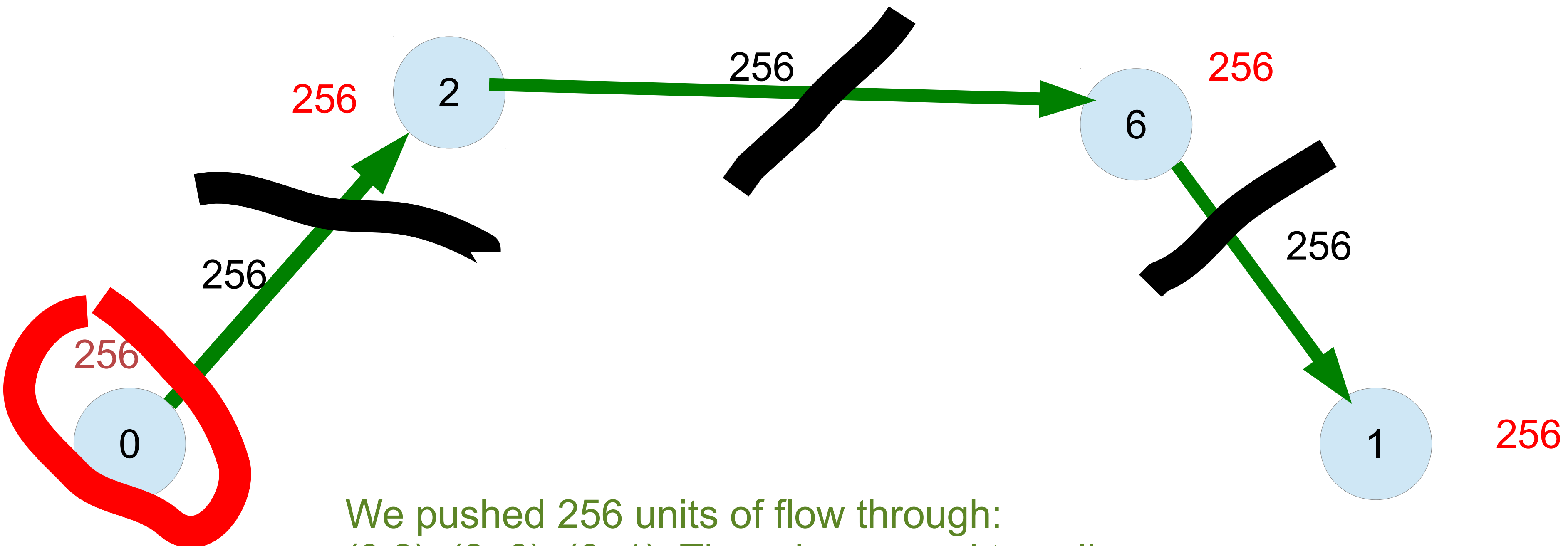
Red = throughput

Pushing from node with min throughput



Pushing 256 units of flow from node 0.

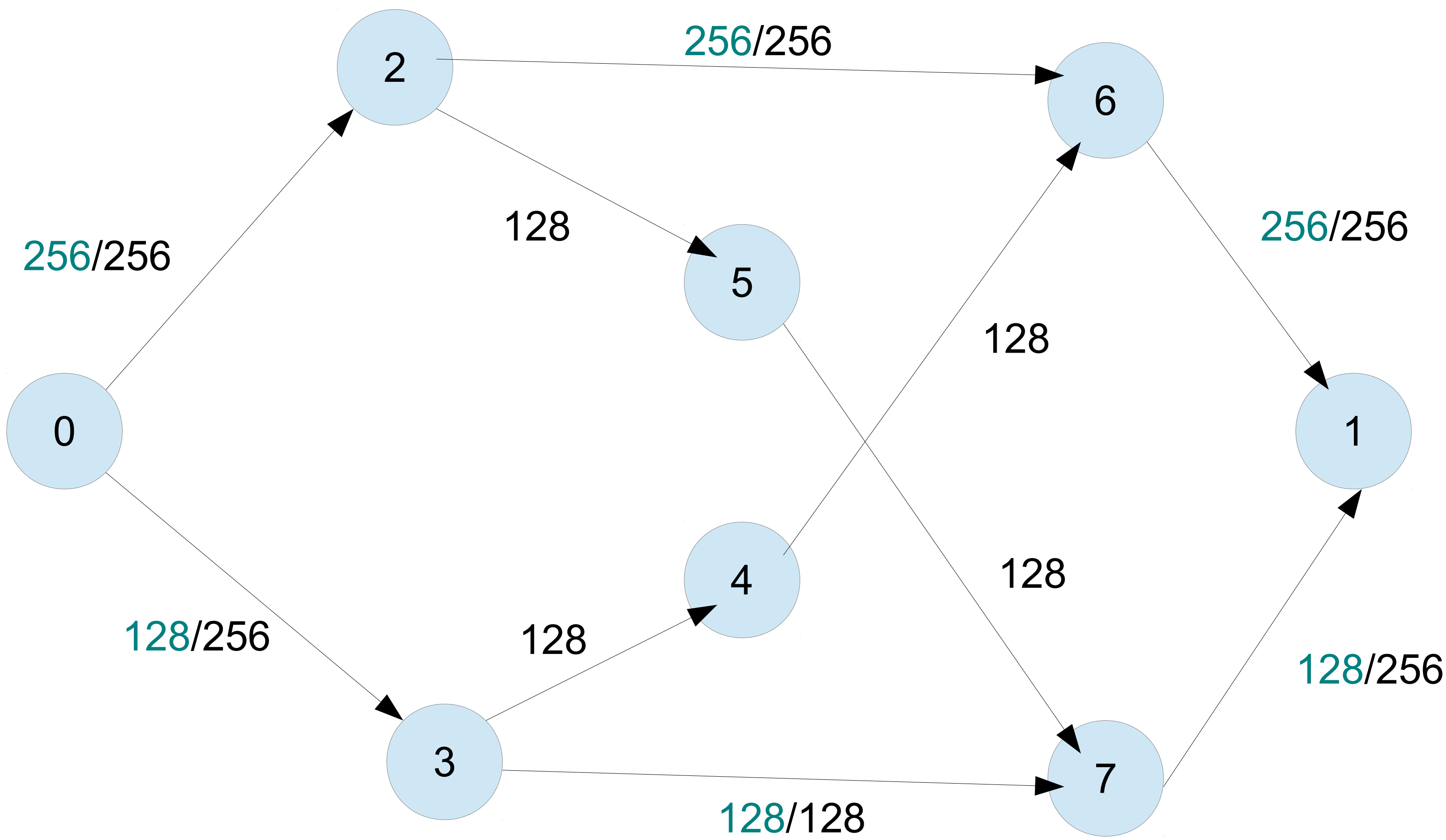
Deleting saturated arcs



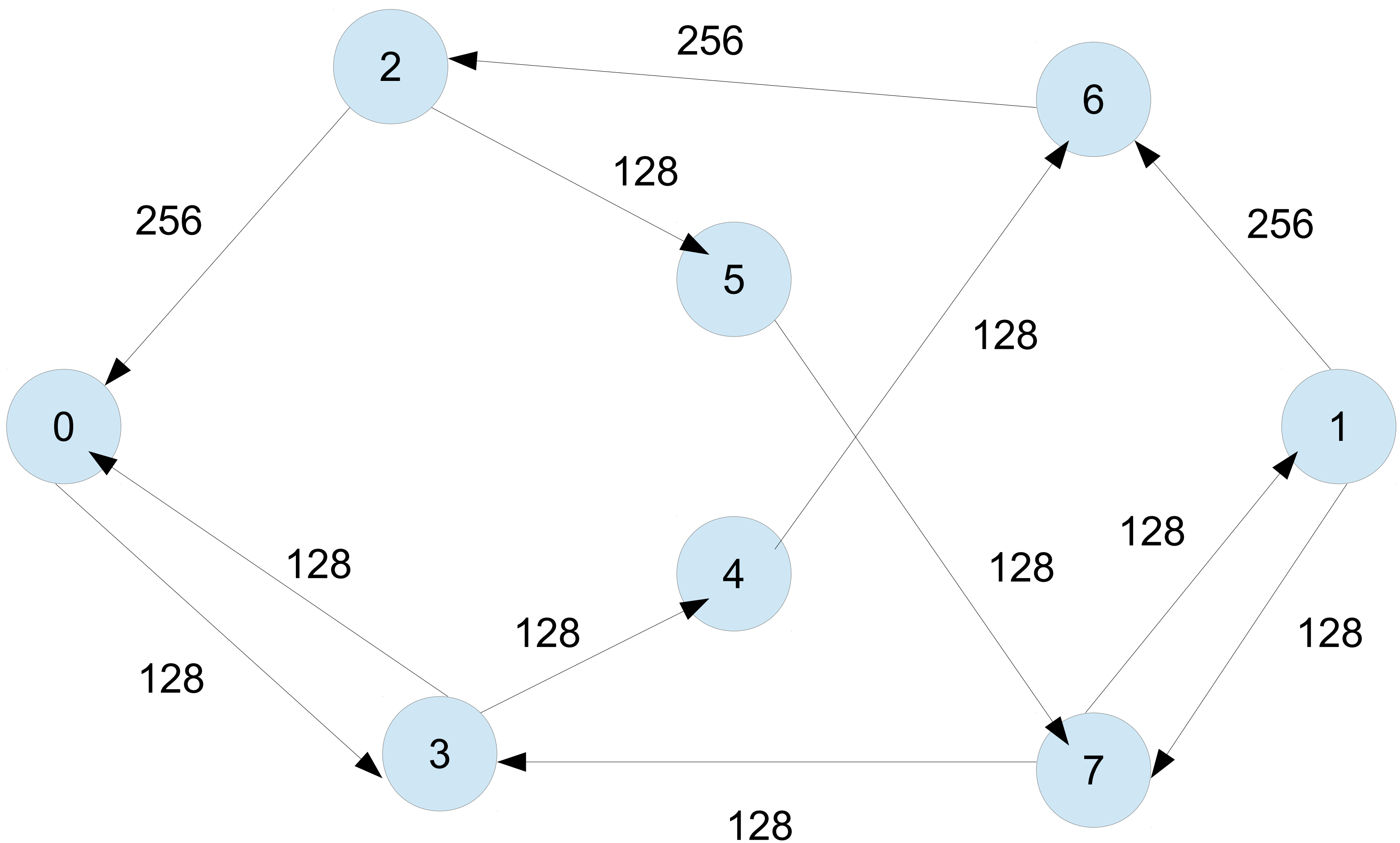
We pushed 256 units of flow through:
(0,2), (2, 6), (6, 1). There is no need to pull
because node 0 was the source node.

Node 1 (sink) is not reachable from 0.
So we have to add the flow to the original
network and start again by building the
new auxiliary.

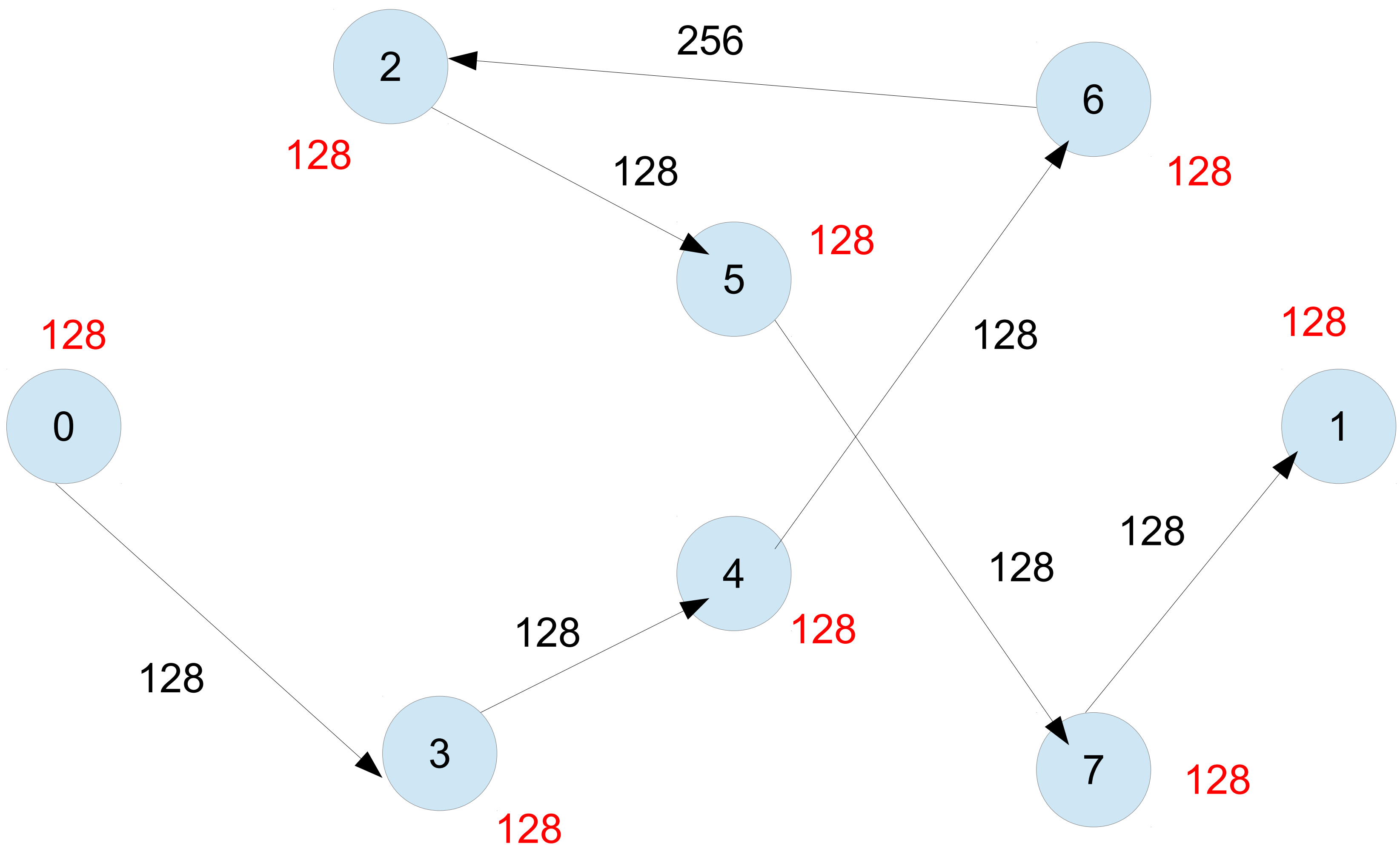
Updating flows in original network



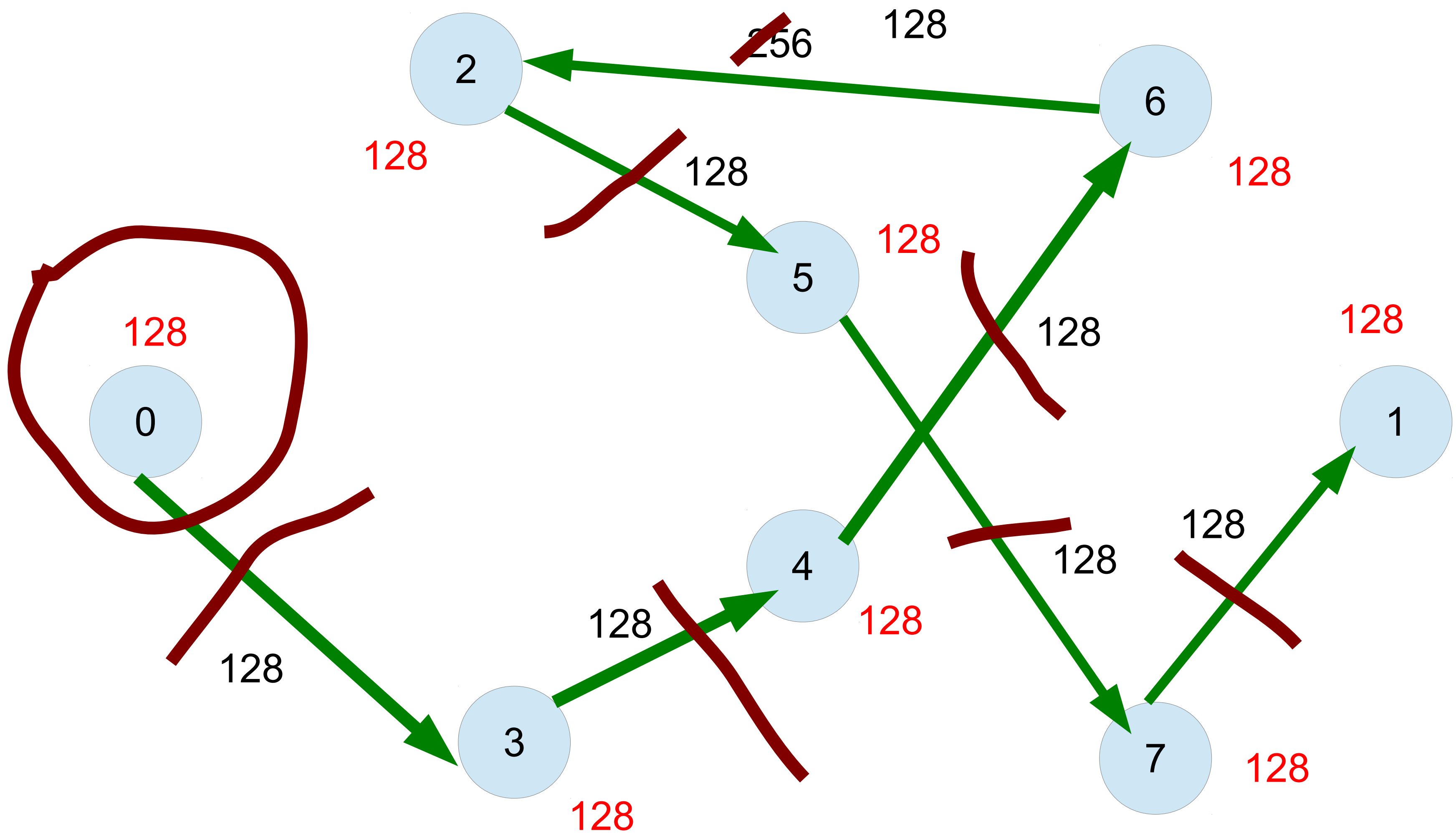
Building residual network



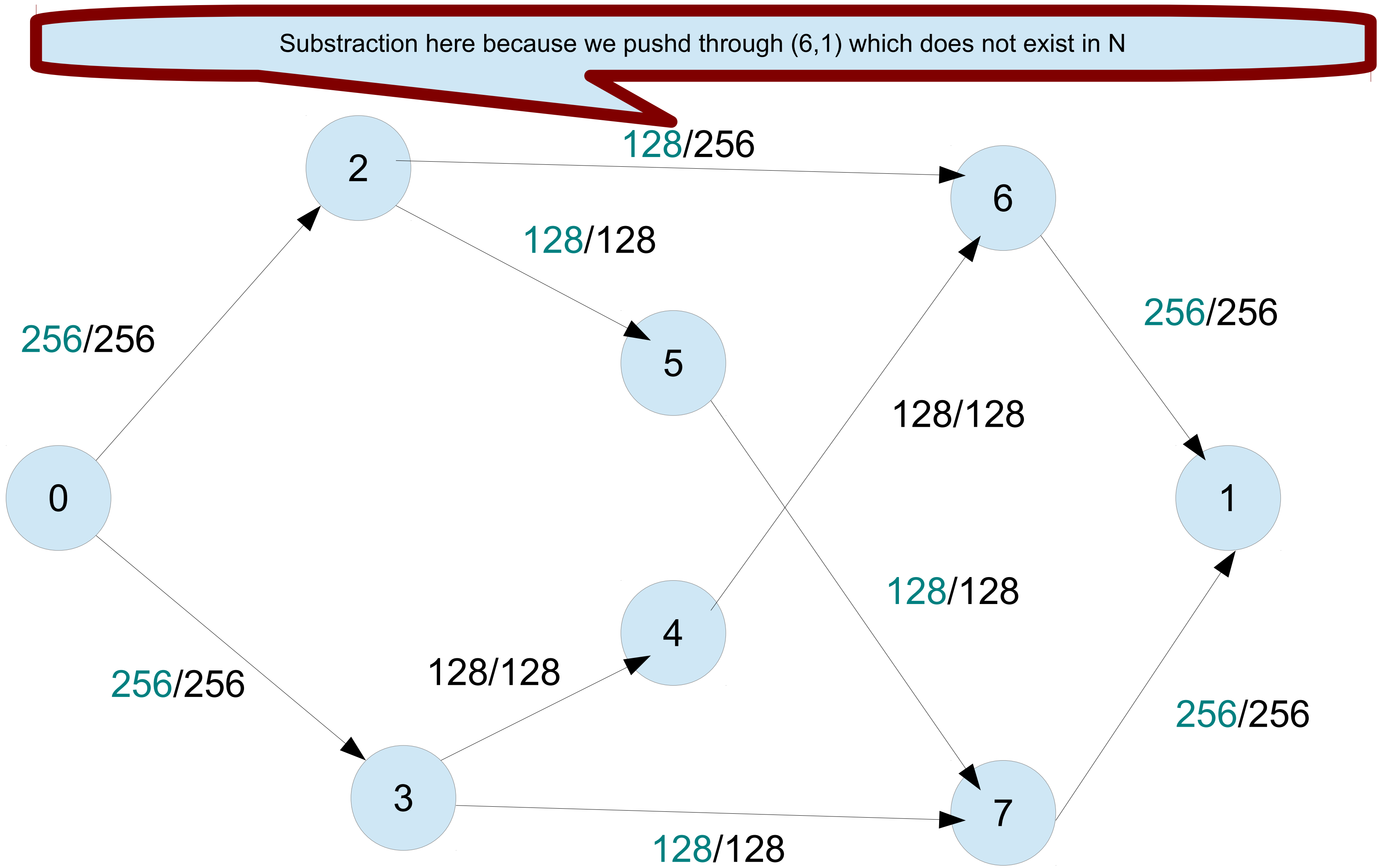
Building auxiliarry network



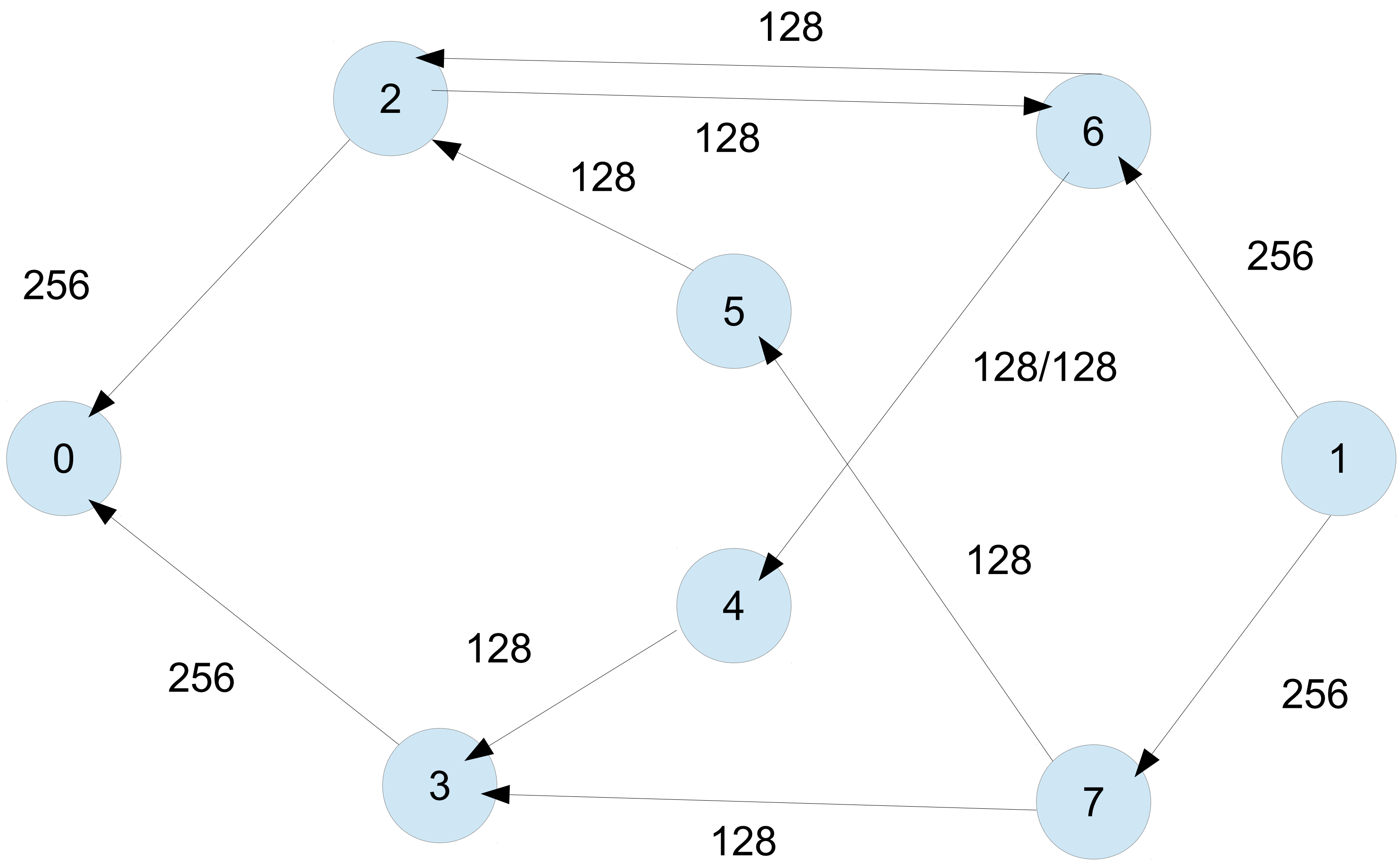
Pushing from node with min throughput



Updating flows in original network



Building residual



Building Auxiliary Network

- We create it while carrying out the breadth first search on the residual network by keeping only the arcs that leads us to new nodes and only the nodes that are at lower level than t.
- BUT we have finished since sink node (1) is not reachable from source node (0) in the residual network.