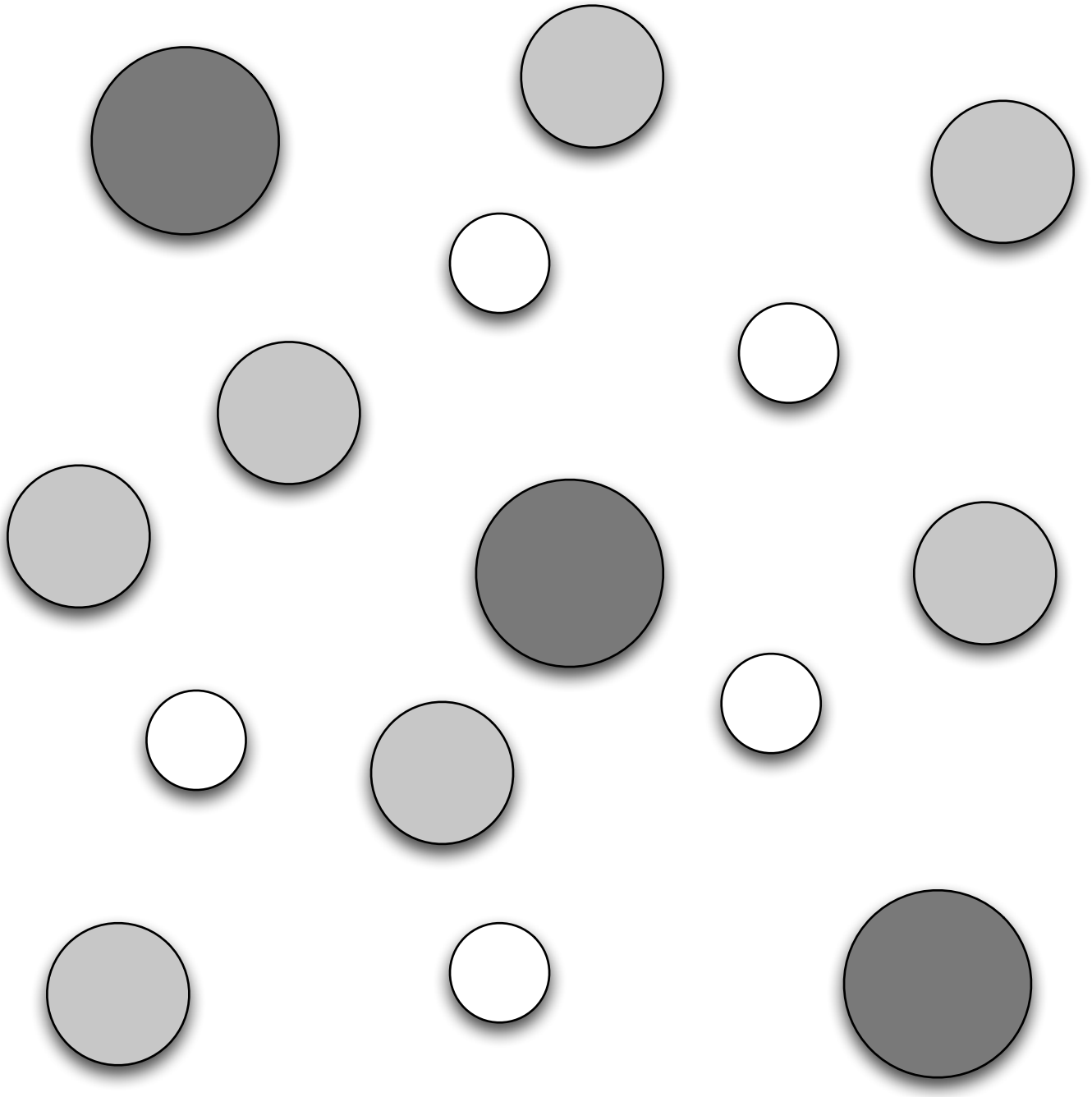
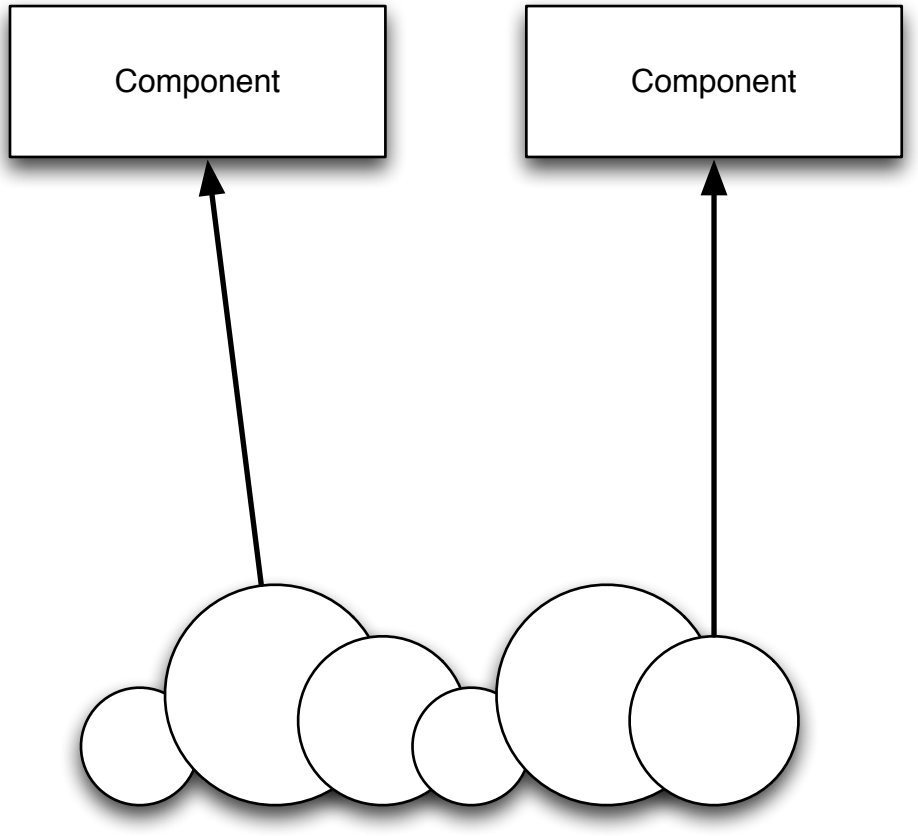
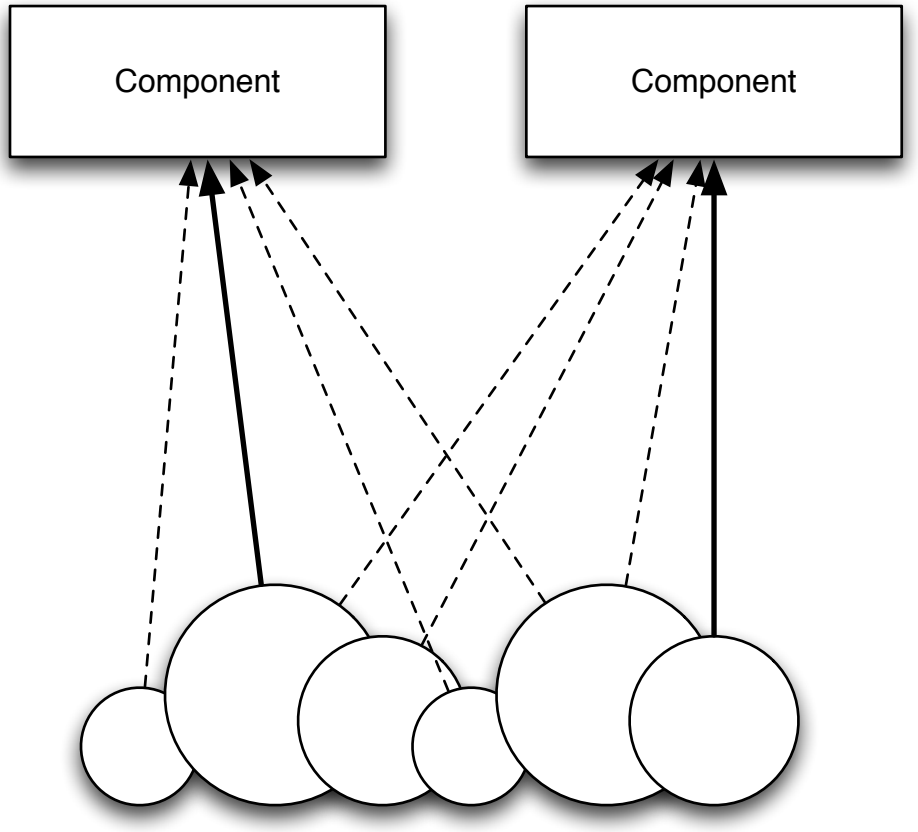
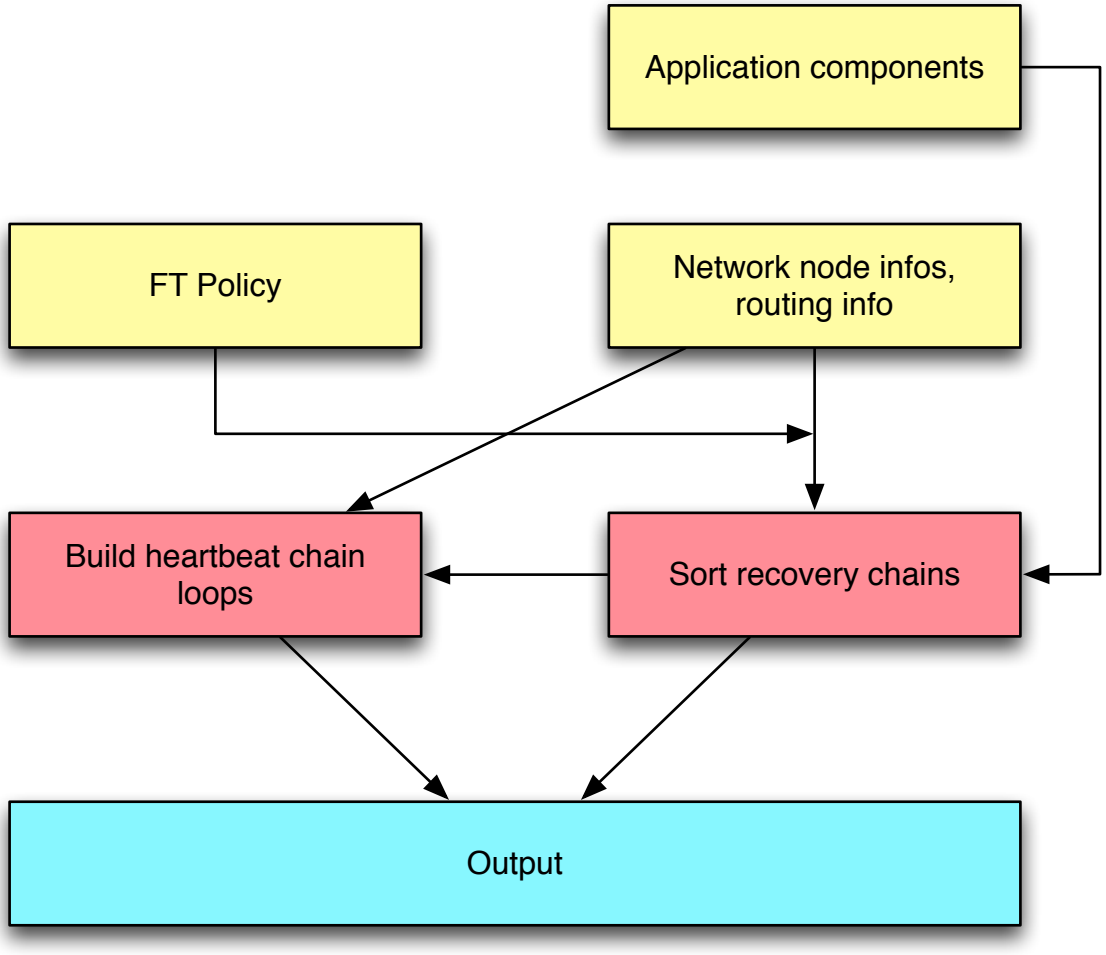
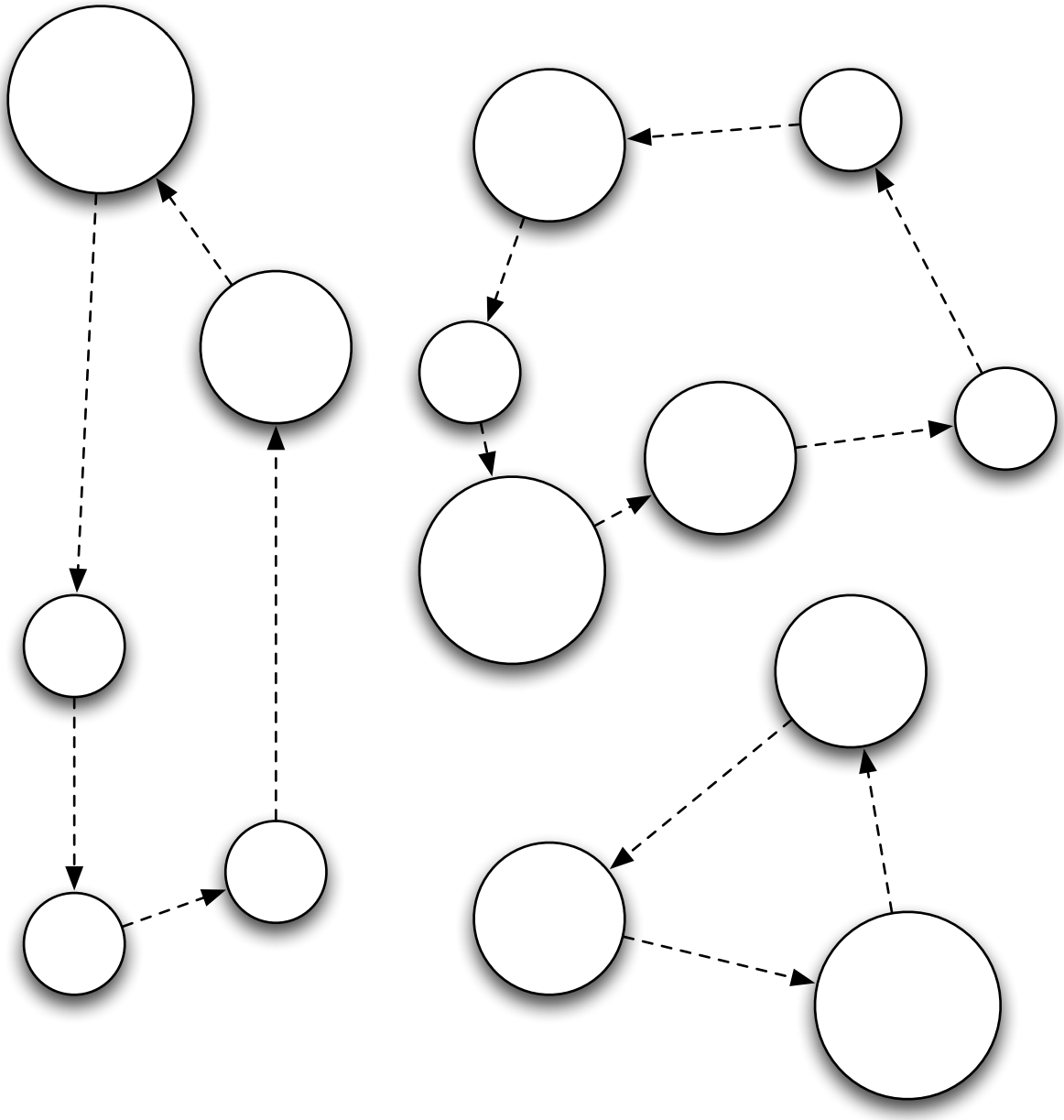

Application Flow.xml
Configuration Policy
Master
Mapping compiler
Discovery Result
Map names to IDs
Translate RBF to Java
Sensor Repository
Application.java
Network topology
Java Compiler
Application Bytecode
Upload

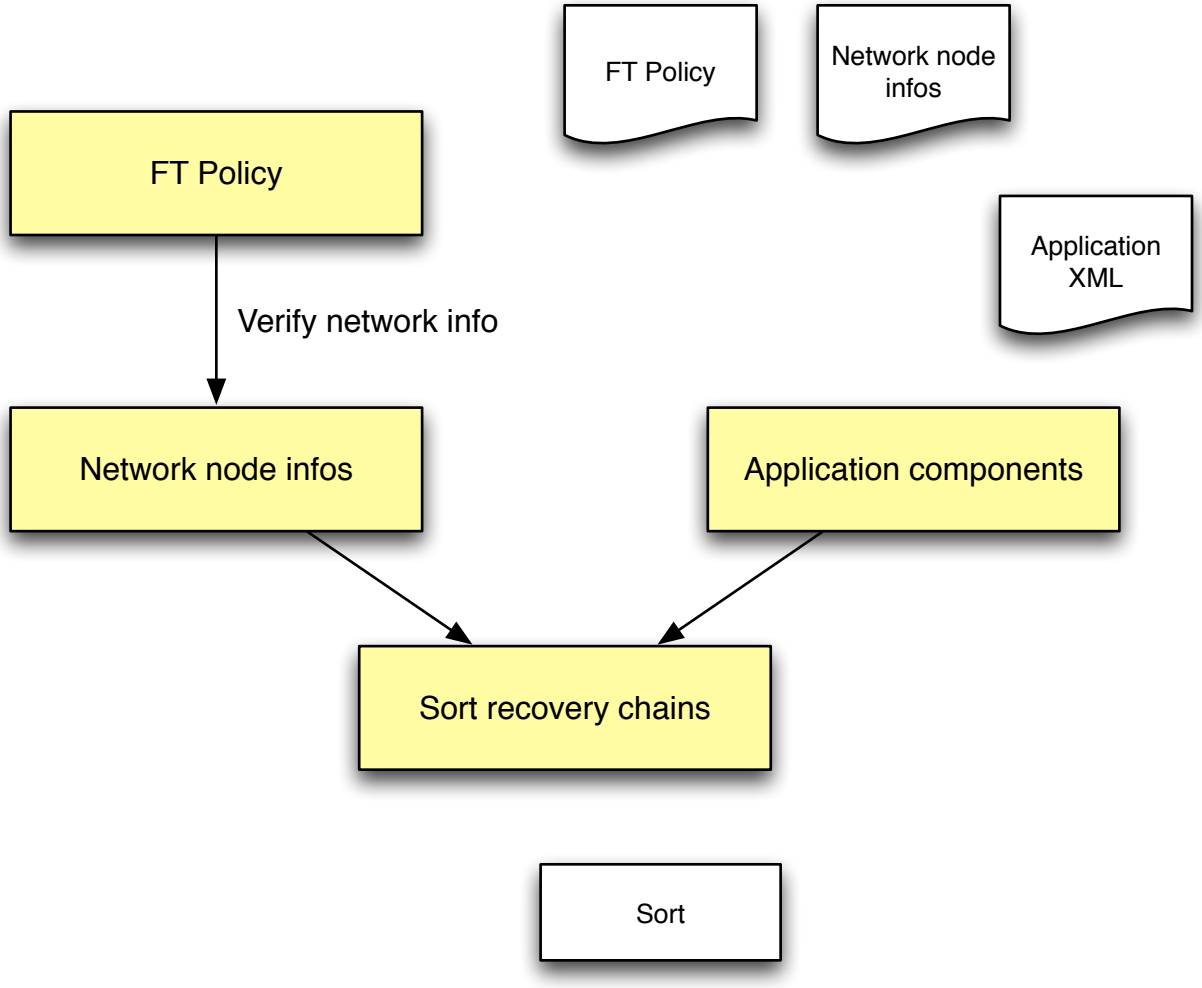


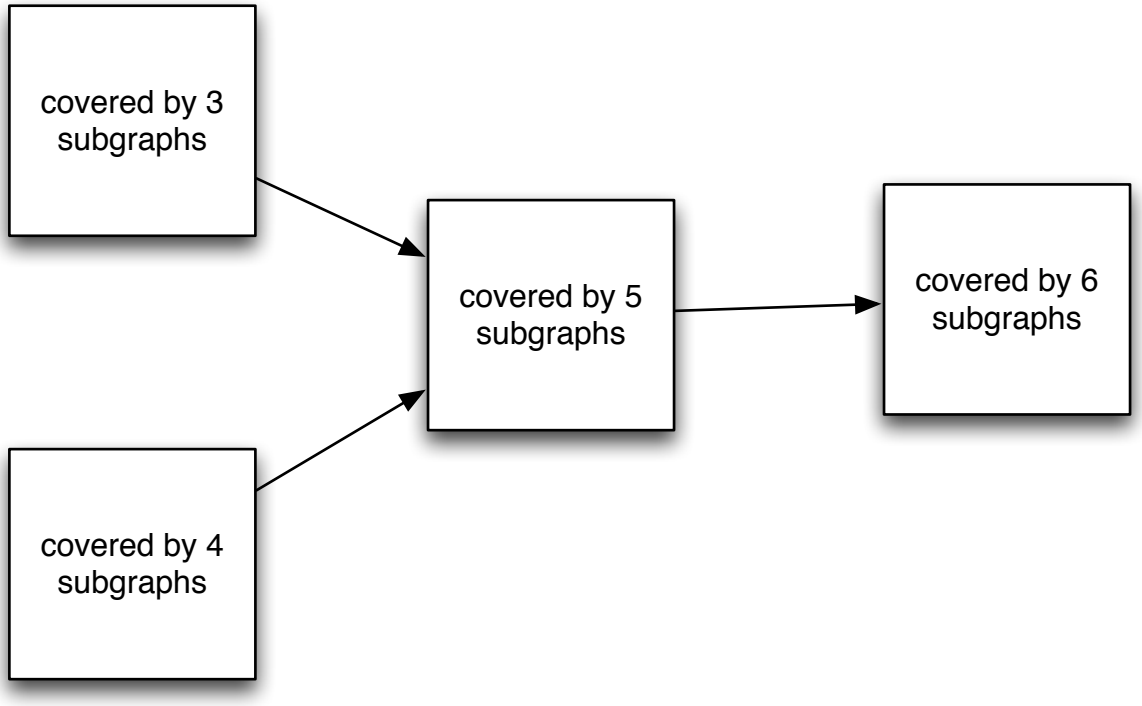

Component
Component


Component
Component


Application components
FT Policy
Network node infos, routing info
Build heartbeat chain loops
Sort recovery chains
Output

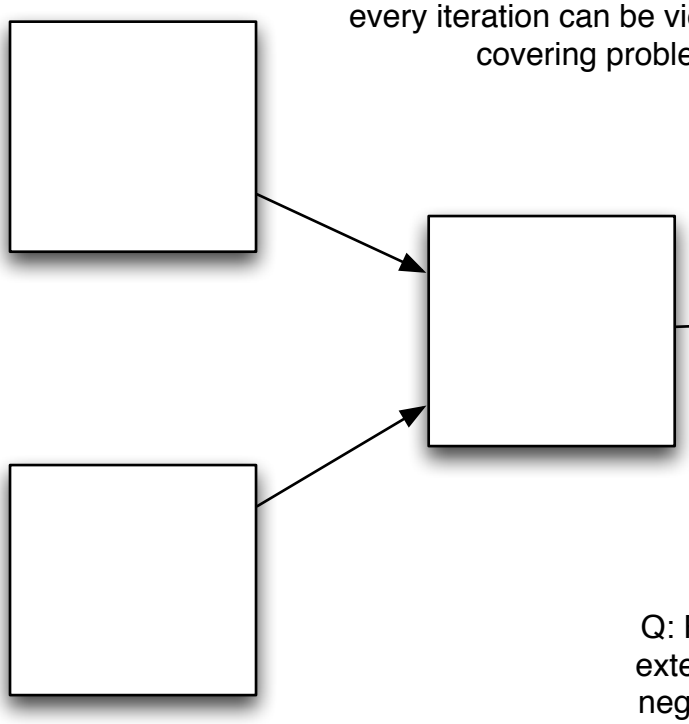



FT Policy
Network node infos
Application XML
FT Policy
Verify network info
Network node infos
Application components
Sort recovery chains
Sort



Assuming each vertex is one component in application
The failure number of a vertex that would render the whole graph not functional if it runs out to 0
is given by the number of subgraphs which include the vertex.

The redundancy level corresponding to the component in FT policy imposed a minimum to this number



every iteration can be viewed as a set covering problem??

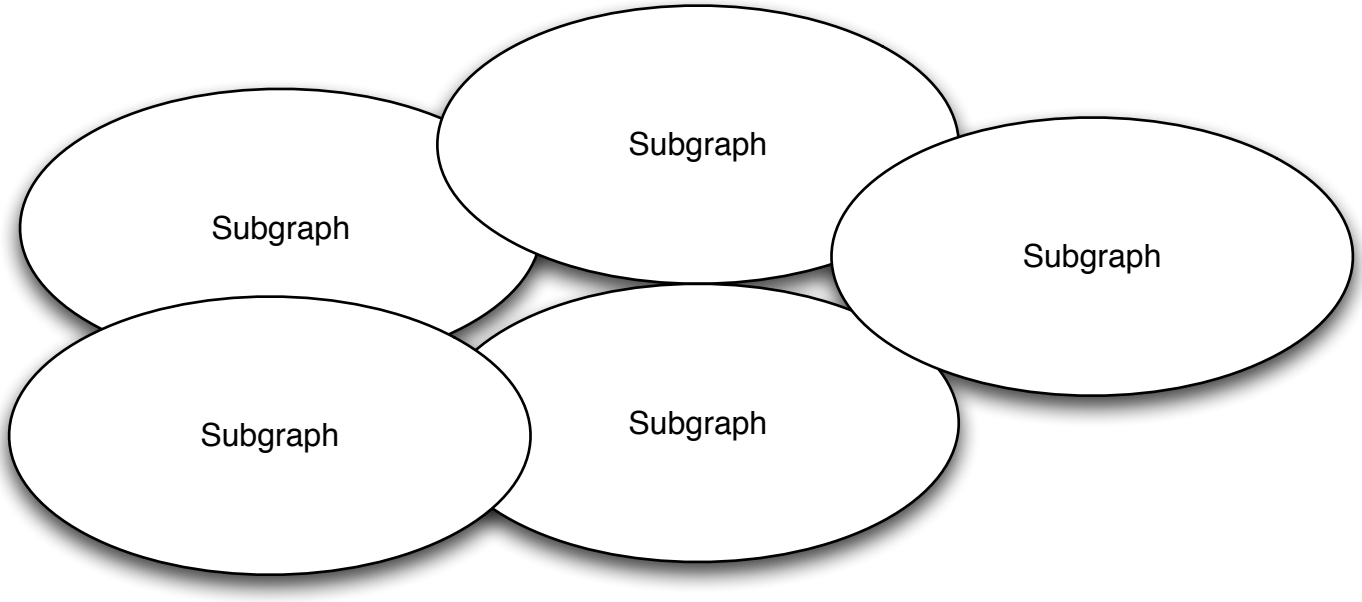
assuming the union of all subgraphs is equal to the graph

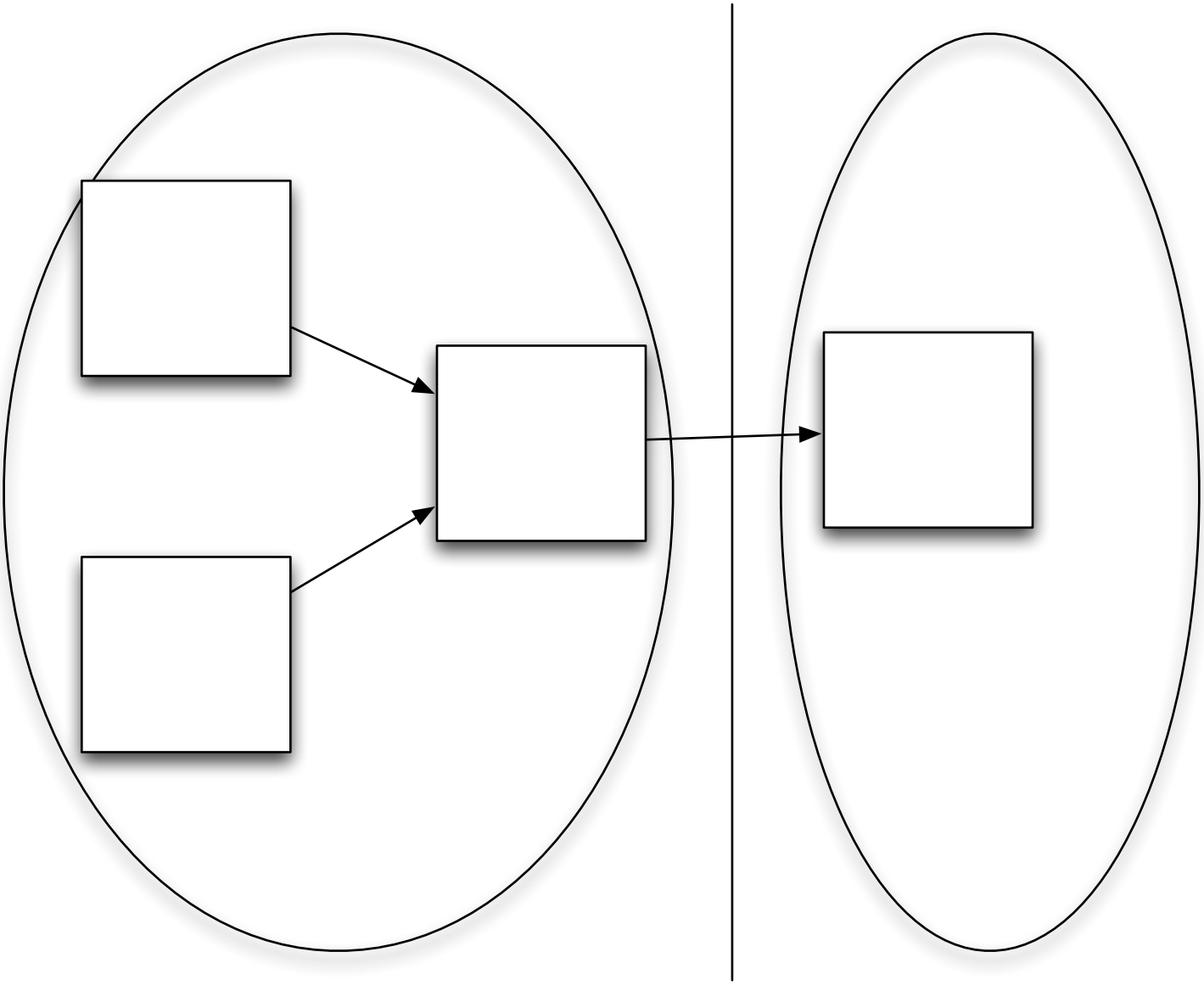
assuming there is zero or finite number of the same subgraph

assuming no empty subgraph

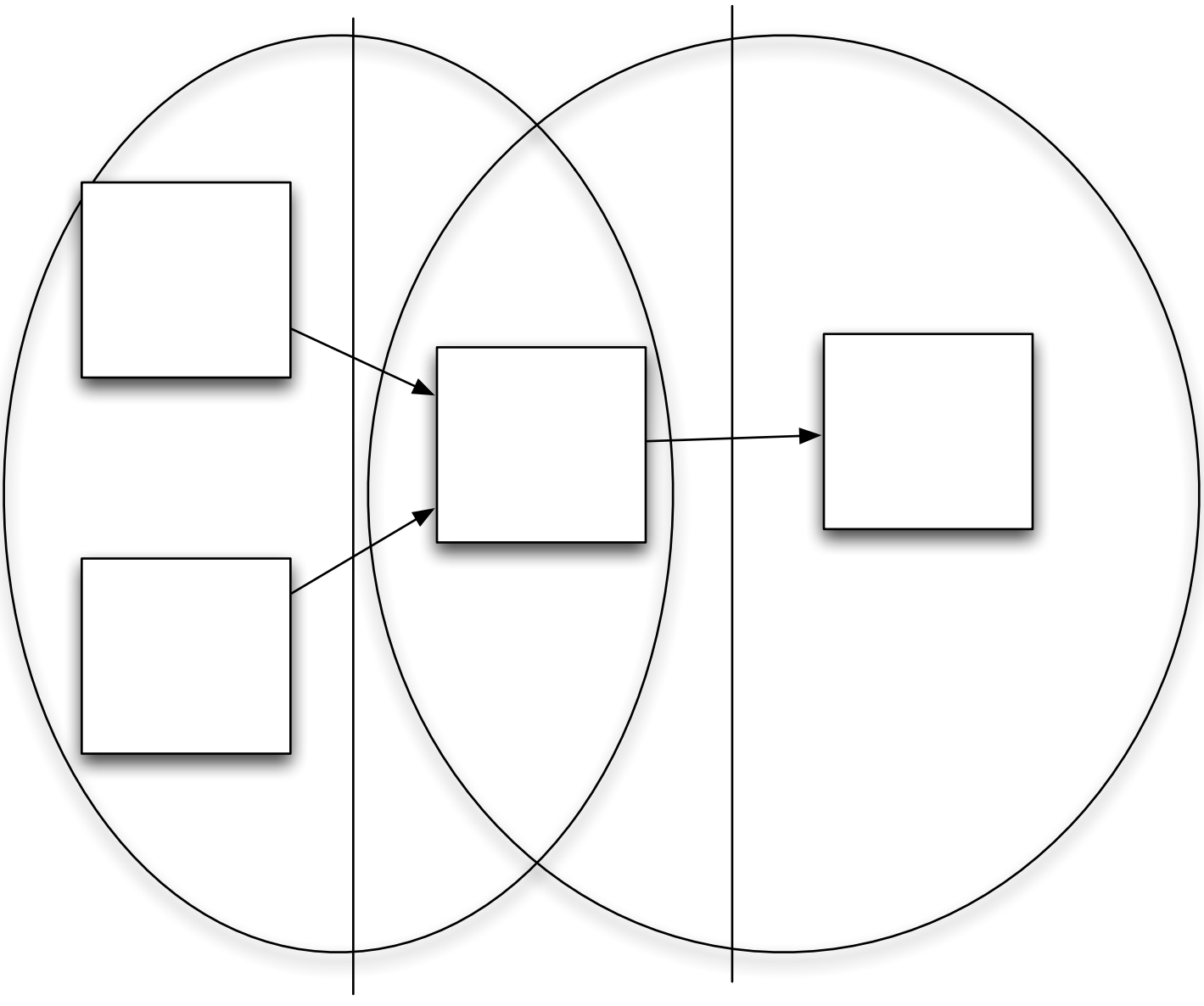
subgraphs could be a subgraph of other subgraphs

Q: Find a list of graph covers with external links (cuts) less than non-negative k for the minimum failure number in all vertices





cut of value 1



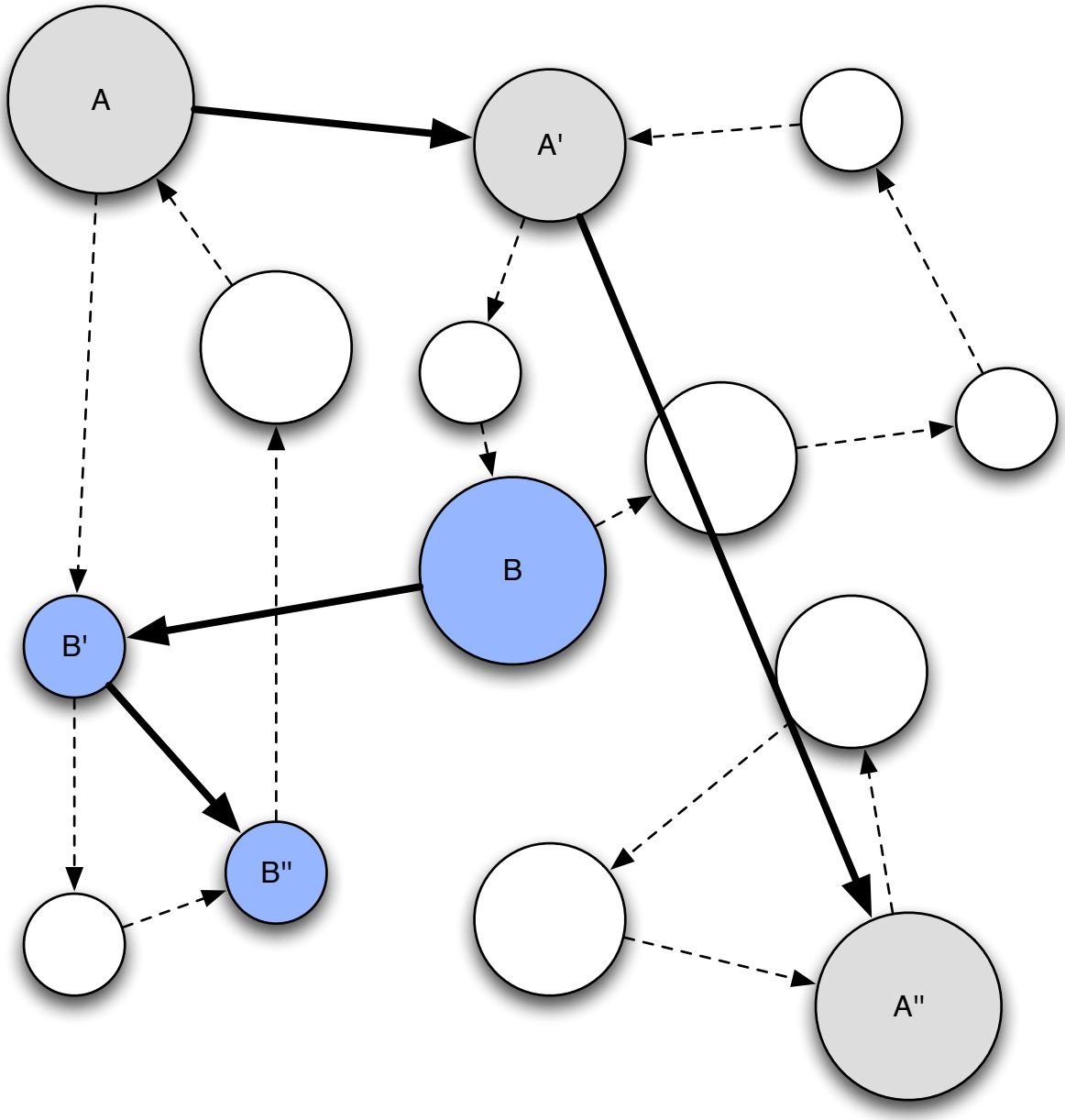
cut of value 2 or 1 (depending on which subgraph the intersected element is in)

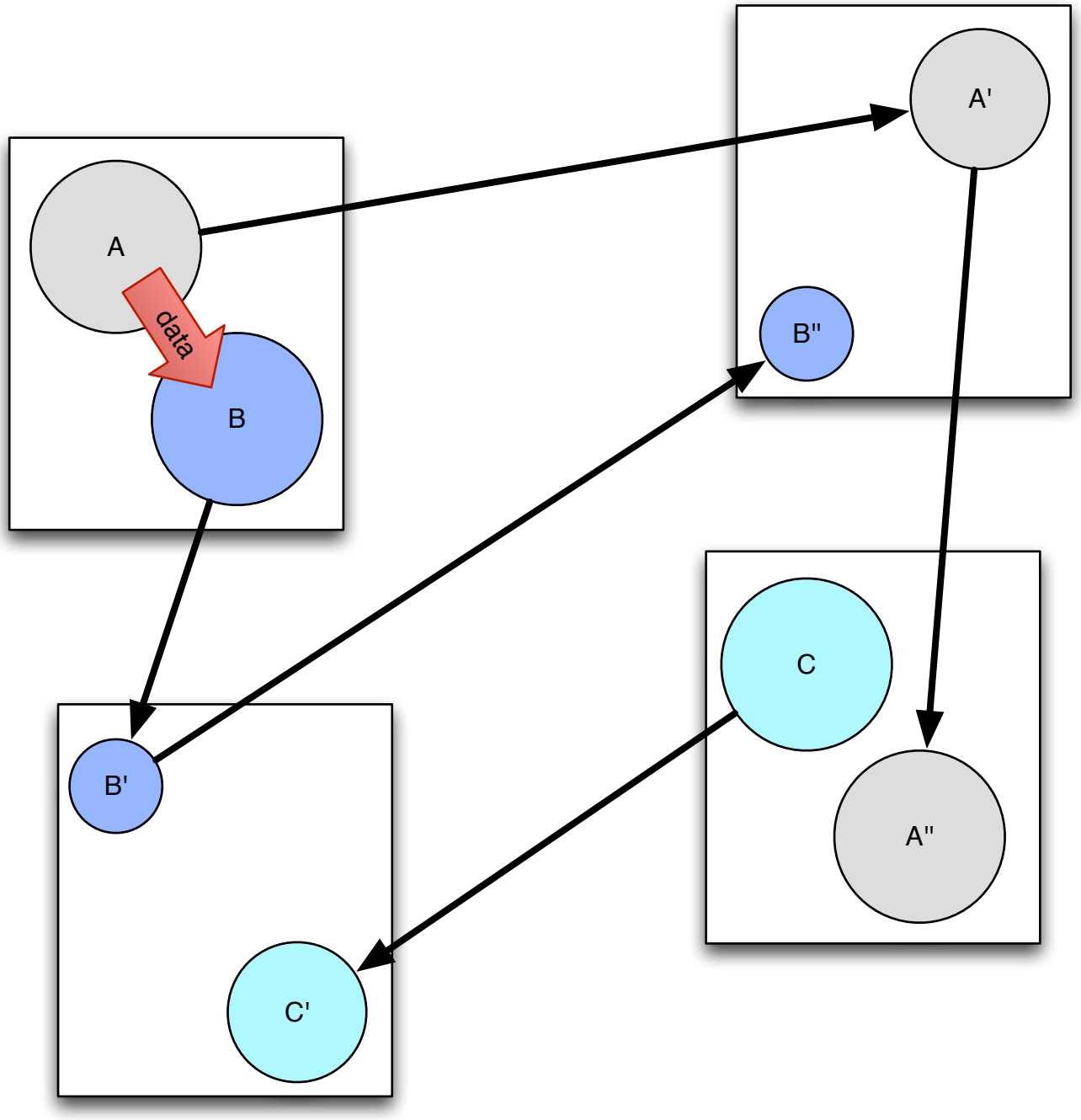
Instead of finding the optimal solution, try to find the first viable one by relaxing the constraints and then find other matches with reasonable failure patterns as well

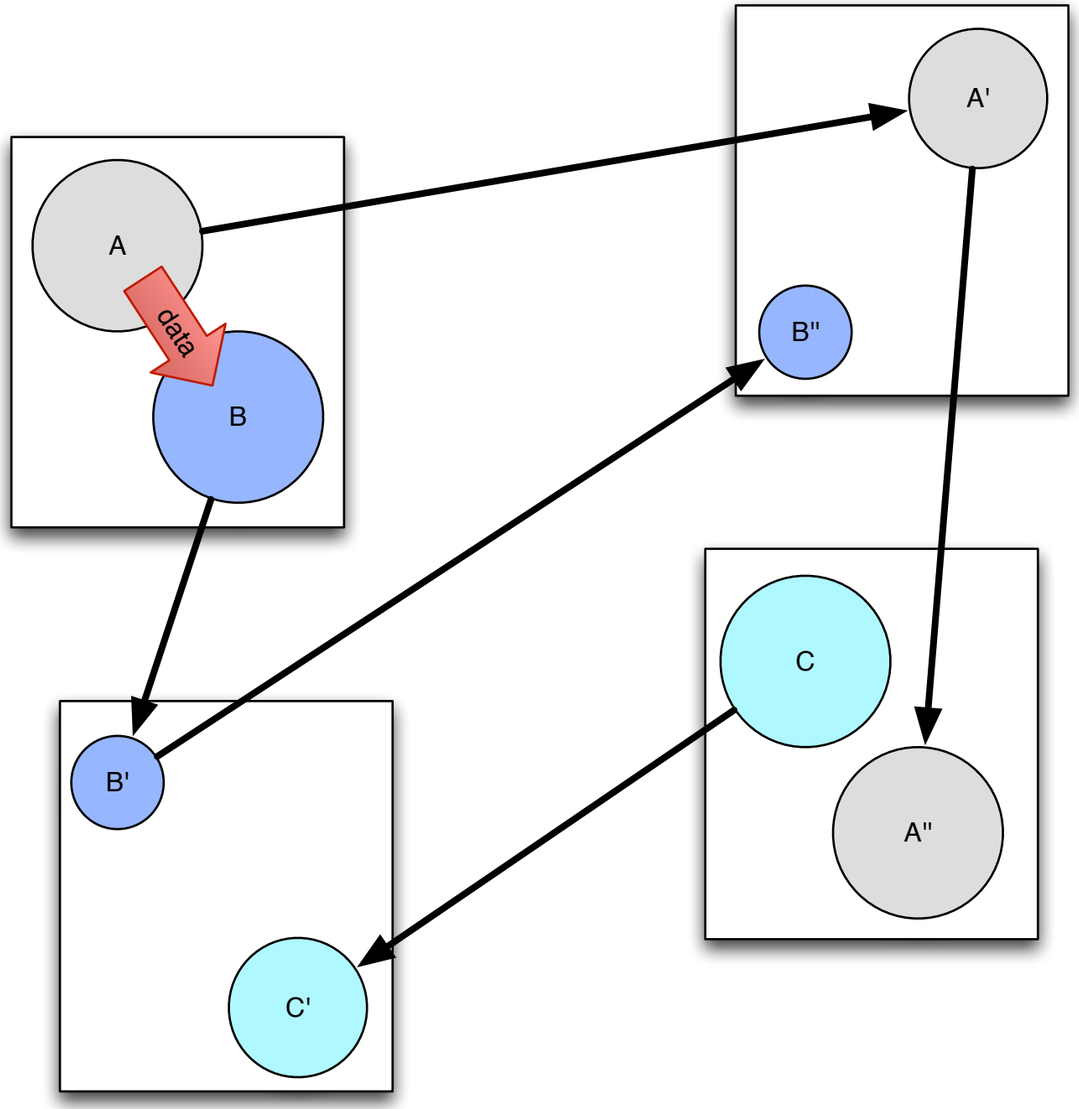
Run greedy approximation, find a set cover, and for every new set find to cover the uncovered set, determine the minimum number of cuts the new set would bring, and once determined which subset to keep the intersection, add the intersection to the list, and remove it from unselected subsets, the differences each keep its node number.

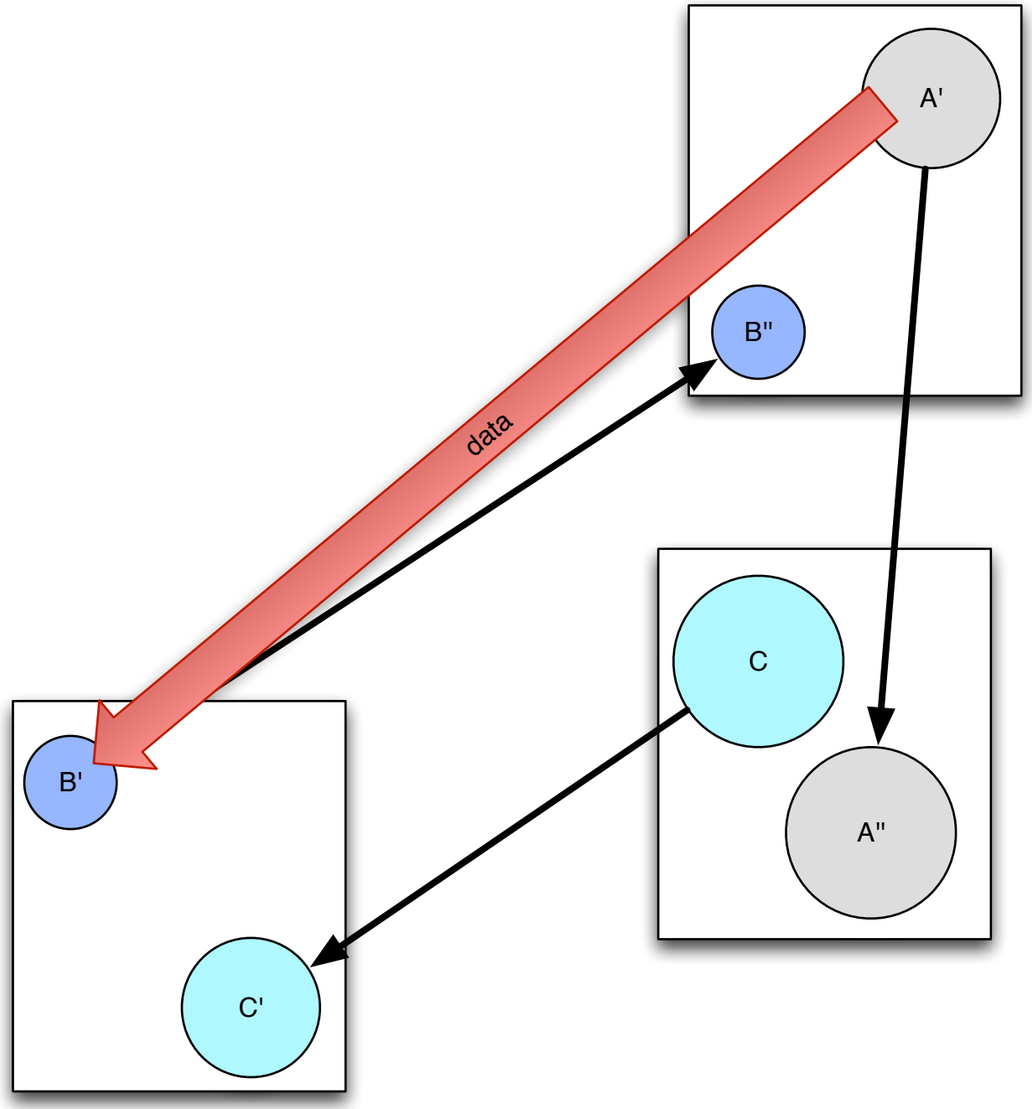
And once a cover is found, remove them from the list and repeat this algorithm until either no set cover exist or no elements left in the list

The greedy approximation is a H_n greedy approximation algorithm
Complexity of algorithm to determine number of cuts is O(n)
The reduction is also in polynomial time


A
A'
B
B'
B''
A''


A
data
B
A'
B"
B'
C
A"
C'


A
data
B
A'
B"
B'
C
A"
C'


A'
B"
data
B'
C
A"
C'