

SOFA Mesh Topology Management

H. Delingette

Mesh Topology

- A Mesh is useful for :
 - Collision detection
 - Haptic rendering
 - Visual rendering
 - Mechanical Modeling (Deformation)
 - Modeling of temperature, electric potential,..

Computational Mesh



Geometry vs Topology

- A mesh is composed of :

- A set of DOFs (Degrees of Freedom) ,
e.g. positions of each node

Geometry
Description

- A description of how those DOFS are
connected,
e.g. edges, triangles, tetrahedra

Topology
Description

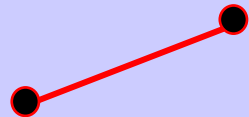
- Topology description is independent of
geometry



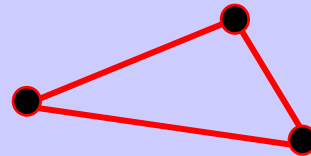
Example of Topologies

- 2 common mesh topologies :

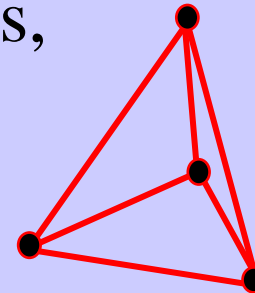
1. Mesh composed of n-simplices,



Edge

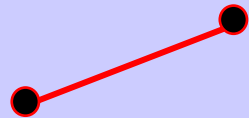


Triangle

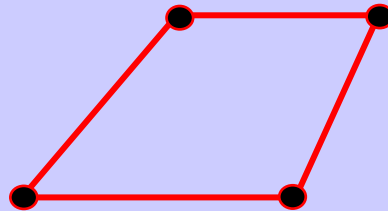


Tetrahedron

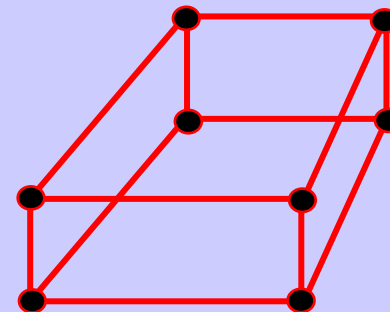
2. Meshes composed of n-cube



Edge



Quad

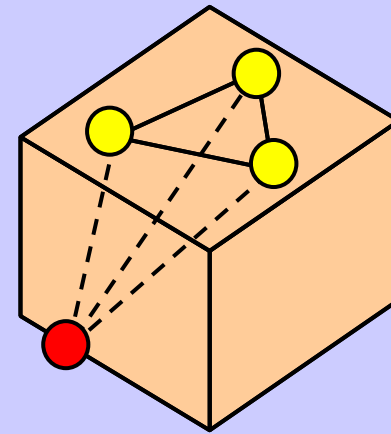
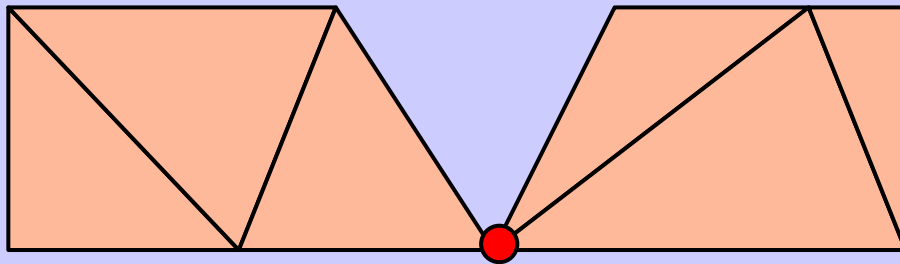


Hexahedron

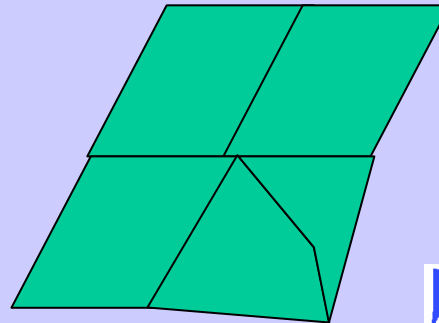
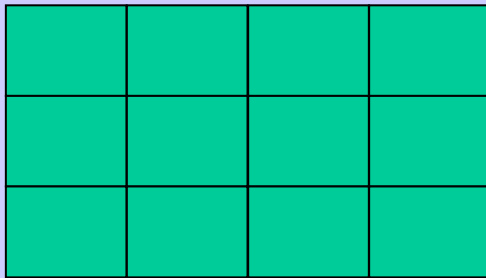


Topology description

- Far more complex classification :
 - Conformal vs Manifold meshes



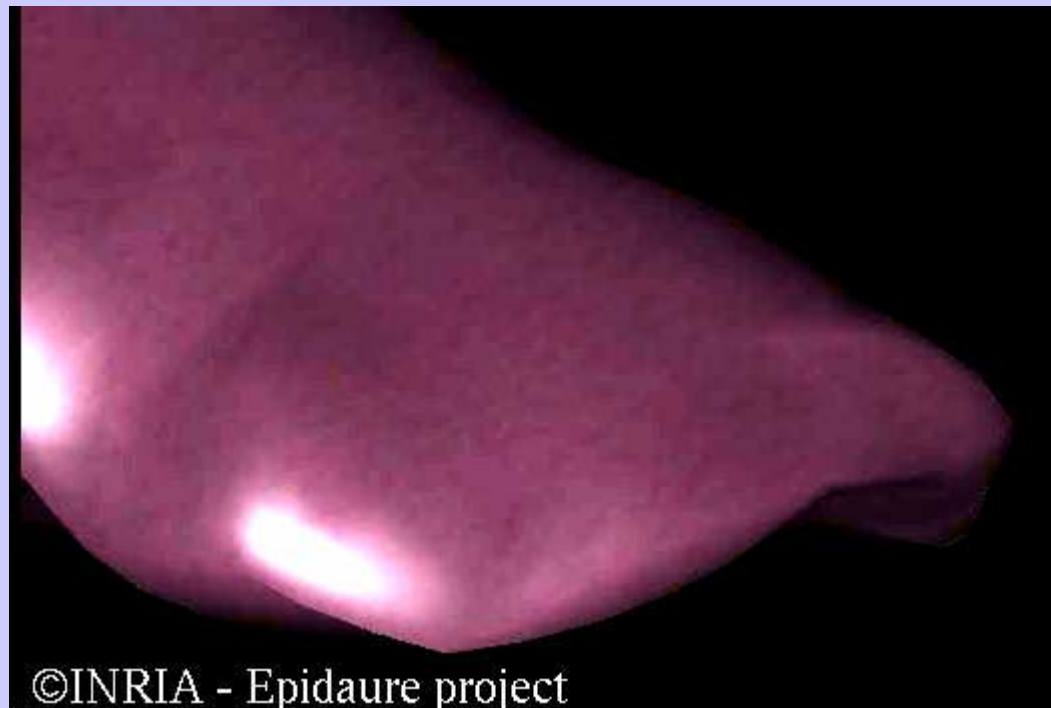
- Structured vs Unstructured grid



Topological Changes

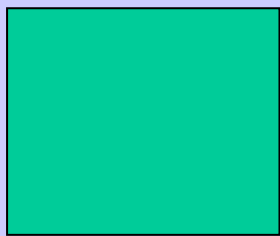
Changing topology is key for medical simulation

Modifying topology entails huge impact for all aspects of the simulation (visual, mechanical, collision detection,..)

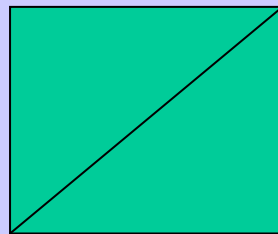


Challenges in implementing Topology Description in SOFA

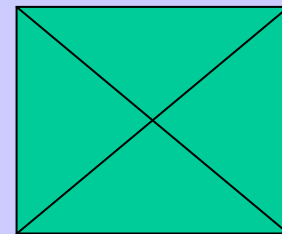
- Provide a generic and hierarchical topology description
- Handle topological changes in a way which is as much transparent for the user as possible.
- Allow multiple topologies for the same DOFs



A quad



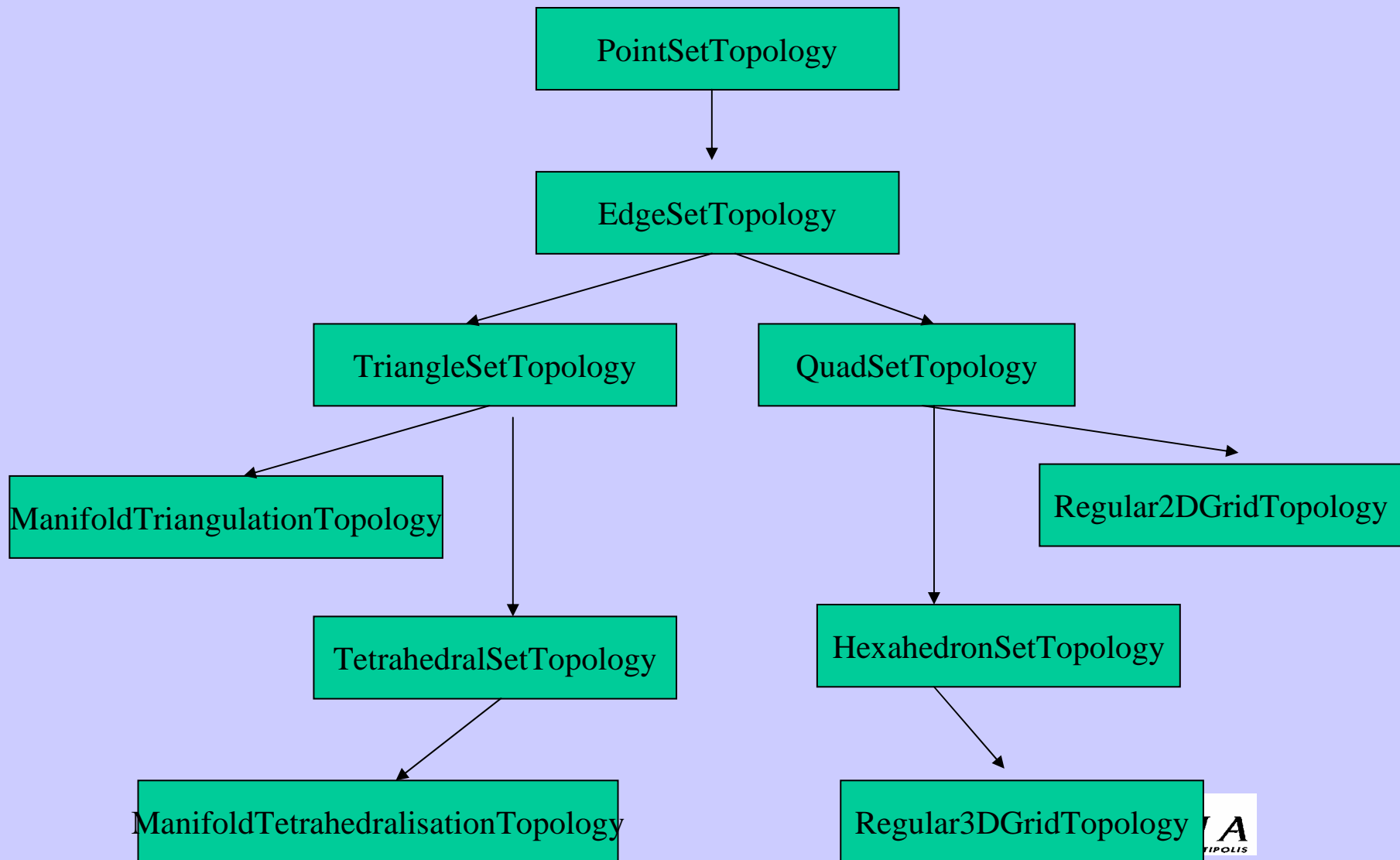
A quad
Decomposed into 2
triangles for FEM
models



A quad
Decomposed into 6
edges for spring-
mass models



Hierarchy of Topologies



Implementation in SOFA

- Currently 2 implementations coexists:
 - MeshTopology class : only provides rough topological description. Not handling topological changes or geometric computation
 - BasicTopology class : new implementation that handles topological changes, full topological relationships and geometric computation.



Definition of a topology

- A BasicTopology object has 4 components :

```
Class basicTopology<DataTypes> {  
  TopologyContainer *container; /// where info are stored and methods to access  
  adjacency  
  TopologyModifiers<DataTypes> *modifier /// low-level methods to change topology  
  TopologyAlgorithms<DataTypes> *topologyAlgorithms /// high-level methods to  
  change topology (user access)  
  GeometryAlgorithms<DataTypes> *geometryAlgorithms; /// methods to get geometry  
  information  
};
```



Information on components

- Container :
 - Adjacency Information is only computed when needed
 - Non template class
 - Store TopologicalChange list
- Modifier :
 - Cannot be accessed from user
 - Modifier also changes the DOFs in the Mechanical Object
 - Most difficult part to code
- TopologyAlgorithms :
 - Accessed from the user
 - High level algorithms to refine, cut mesh
- Geometry Algorithms :
 - Compute geometric information (normal, curvature, area, length)
 - Defined an interface class to store directly information in an array



Exemples

- 4 such topologies implemented so far :
 - PointSetTopology :
 - Container (useless) : For each point gives its index
 - Modifier : addPoint, removePoint, renumberPoints
 - Geometry : computeCenter, computeRadius, getAABB()
 - EdgeSetTopology (inherited from PointSetTopology)
 - Container : array of edges, array of vertex edge shell
 - TopologyAlgo : addEdges, removeEdges, fuseEdges, splitEdges
 - Geometry : getEdgeLength, getRestEdgeLength



Exemples

- 4 such topologies implemented so far :
 - TriangleSetTopology (inherited from EdgeSetTopology)
 - Container : array of triangles, of vertex and edge triangle shell
 - Modifier : addTriangle, removeTriangle
 - Geometry : computeTriangleNormal
 - TetrahedralSetTopology (inherited from TriangleSetTopology)
 - Container : array of tetrahedra, array of vertex, edge, triangle tetrahedra shell
 - TopologyAlgo : addTetrahedra, removeTetrahedra
 - Geometry : getTetrahedronVolume



Container Data Structures

- Force Fields, Constrains, Mapping may require to store information for each topological item (point, edge,...)
- Defined 2 container classes that handle topological changes
 - `PointData<MyType>`, `EdgeData<MyType>` are arrays (same as `std::vector`) of item of type `MyType`
 - `PointSubset`, `EdgeSubset` are arrays of points or edges
 - There are used-defined functions that are called when an item is created or destroyed



Container Data Structures

- Those container data structures are “aware” of topological changes.
- User only provide callback functions to handle :
 - Destruction of a topological item
 - Creation of a topological item



Exemple : Spring Force Field

- The structure stored for each edge

```
struct Spring {  
    Real ks;           // spring stiffness  
    Real kd;           // damping factor  
    Real restLength;  // rest length of the spring  
    Mat3 dfdx;        // the edge stiffness matrix  
  
    Spring(Real _ks, Real _kd, Real _rl) {}  
    Spring() : ks(1.0),kd(1.0),restLength(0.0) {}  
}
```

- The Force Field contains an EdgeData

```
class SpringEdgeDataForceField {  
    EdgeData<Spring> springArray;  
  
    ....  
};
```

Exemple : Spring Force Field

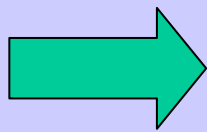
- The creation callback function

```
template<class DataTypes> void springCreationFunction(  
int index, // the index of the edge to be created  
void* param, // a void * (in reality a SpringForceField * object)  
Spring& t, // a reference on the structure to be modified  
const Edge&e , // the edge that has been created  
const std::vector< unsigned int > &ancestors, // an array of edge ancestors  
const std::vector< double >& coefs // for each edge ancestor a weight (all weights sum to 1)  
)  
{  
// initialize the rest edge length of the spring and sets the stiffness and viscosity  
}
```



Other example DiagonalMass

- The mass of the object must be modified during topological changes



Mass should be computed from each element using $\text{MassDensity} * \text{length, surface or volume}$

- Nodal masses are stored in a `PointData` array
- Add creation and destruction callbacks when edges, triangle, tetrahedra are created/removed

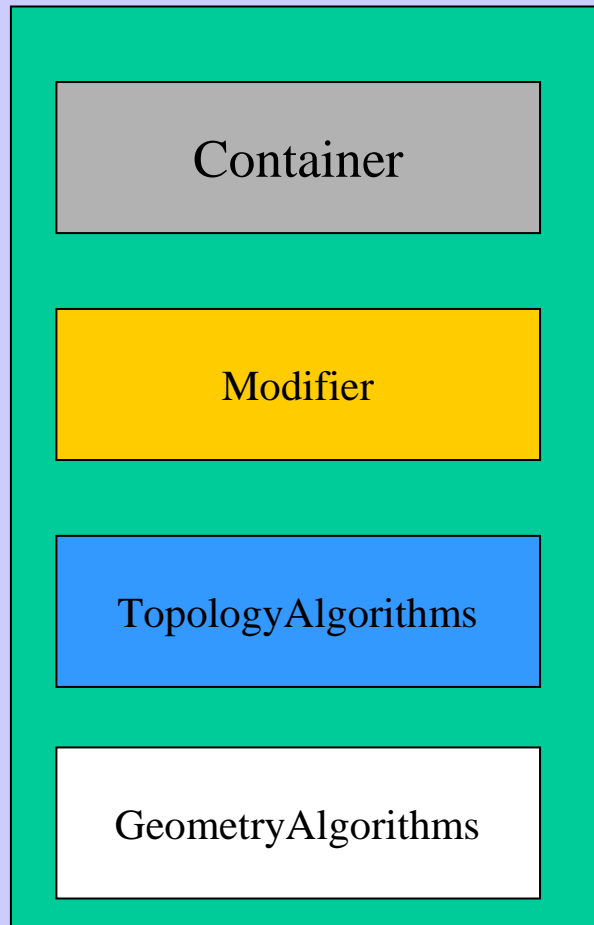


Last example FixedConstraint

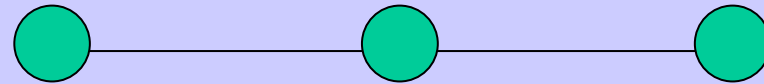
- The list of points having zero displacements are stored in a PointSubset
- Use a callback function when a point is created
 - By default do not add newly created point



What happens when I split an Edge ?



DOFs Mechanical Object

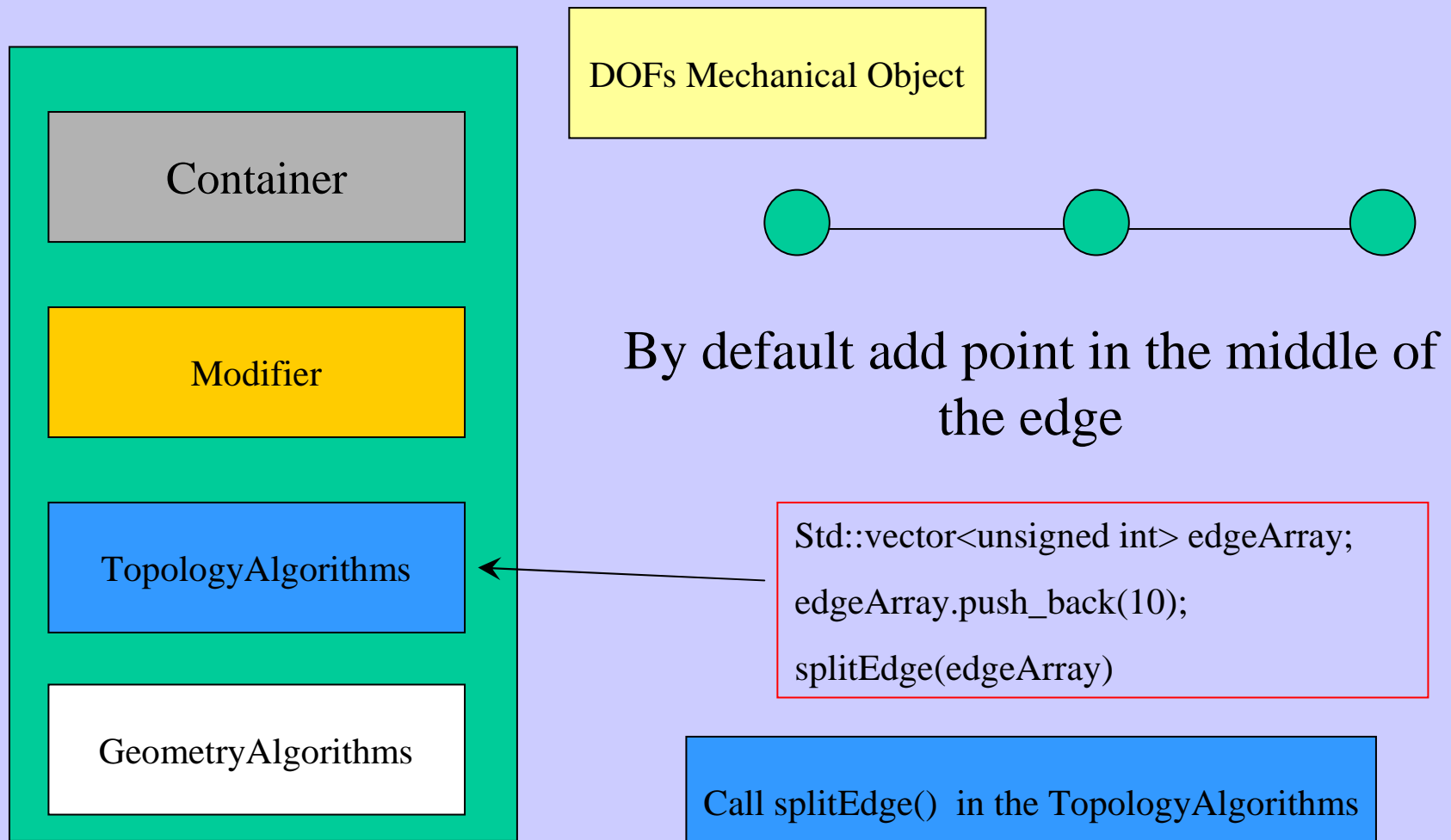


By default add point in the middle of the edge

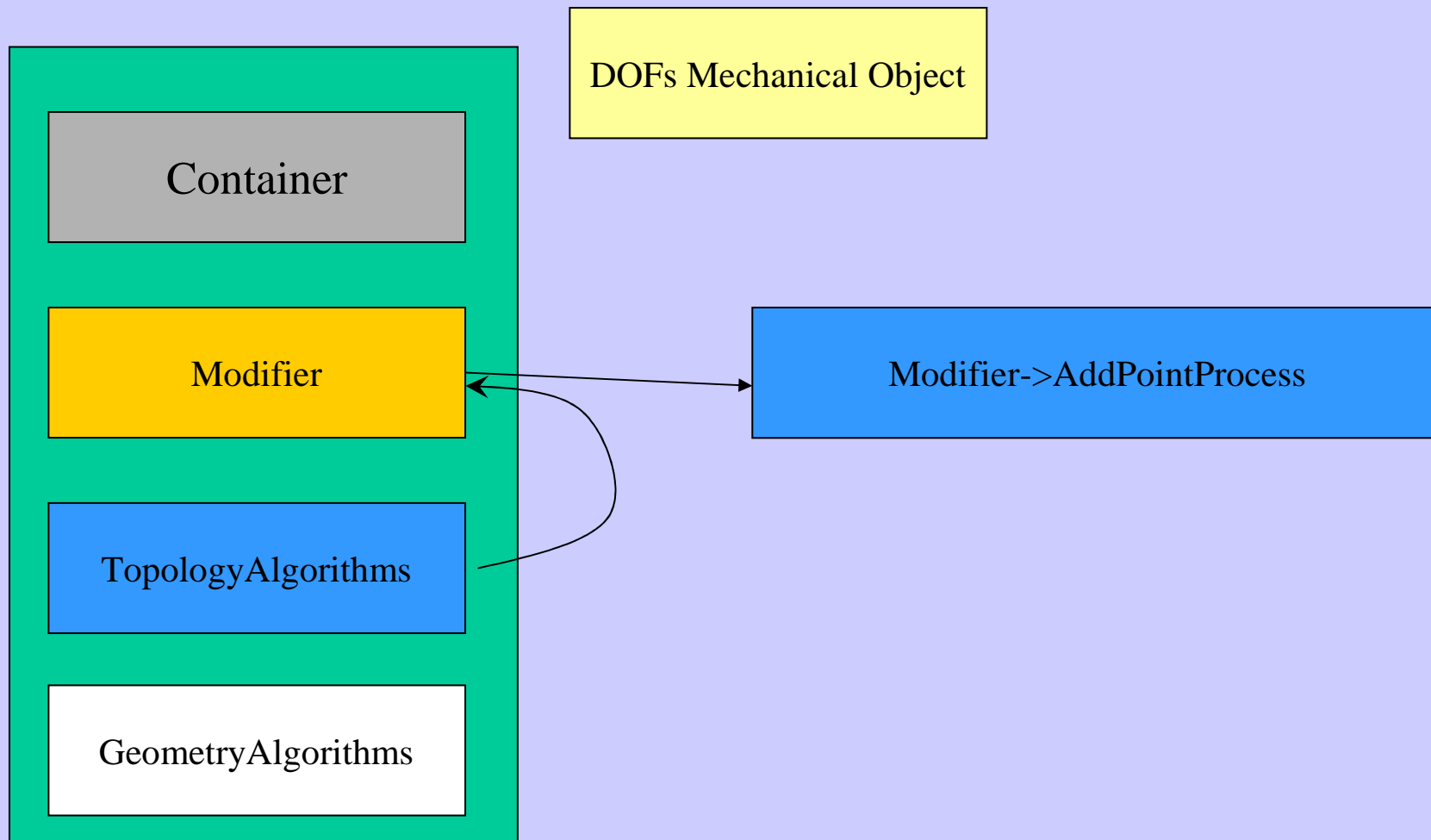
```
Std::vector<unsigned int> edgeArray;  
edgeArray.push_back(10);  
splitEdge(edgeArray)
```



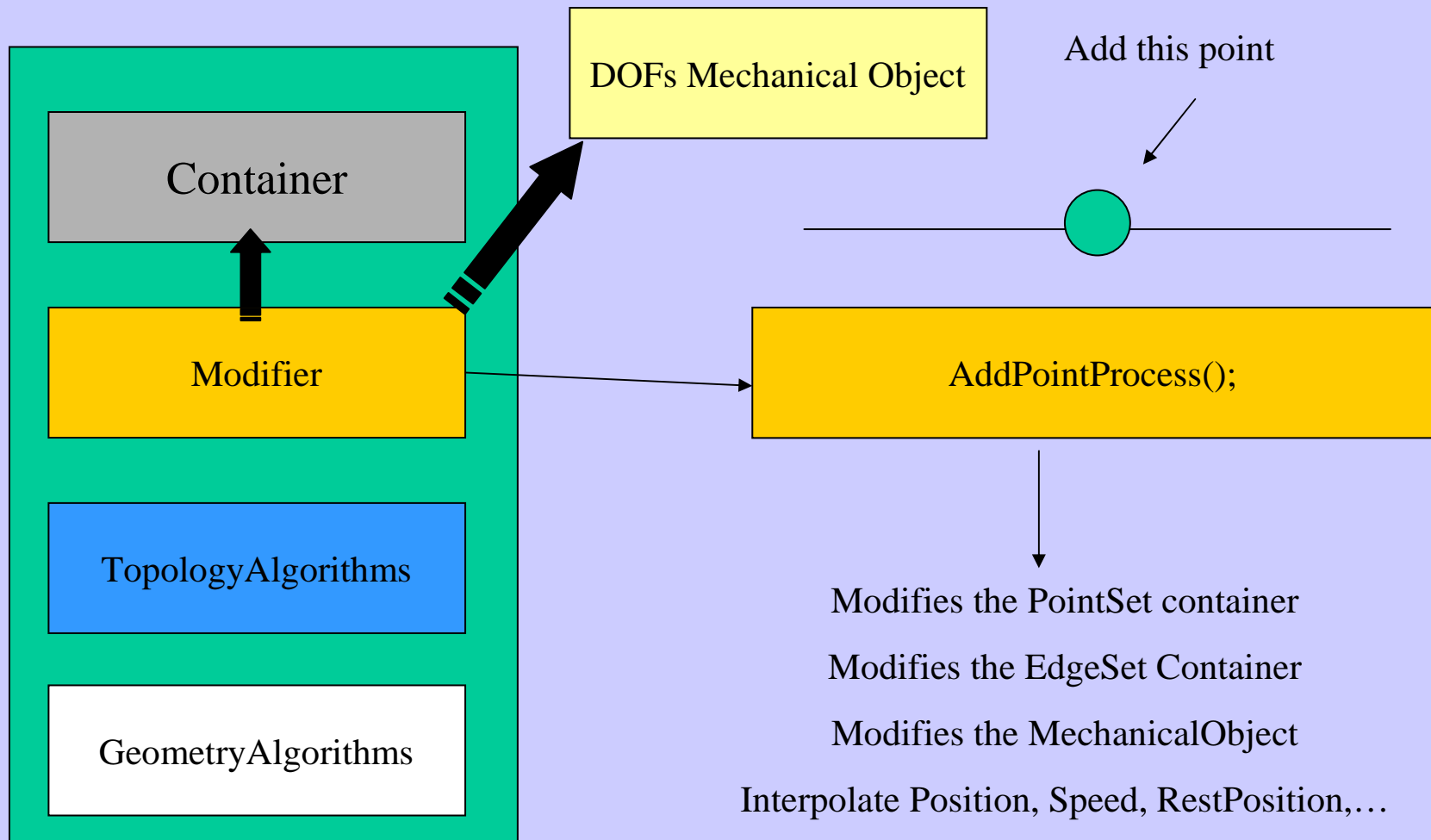
What happens when I split an Edge ?



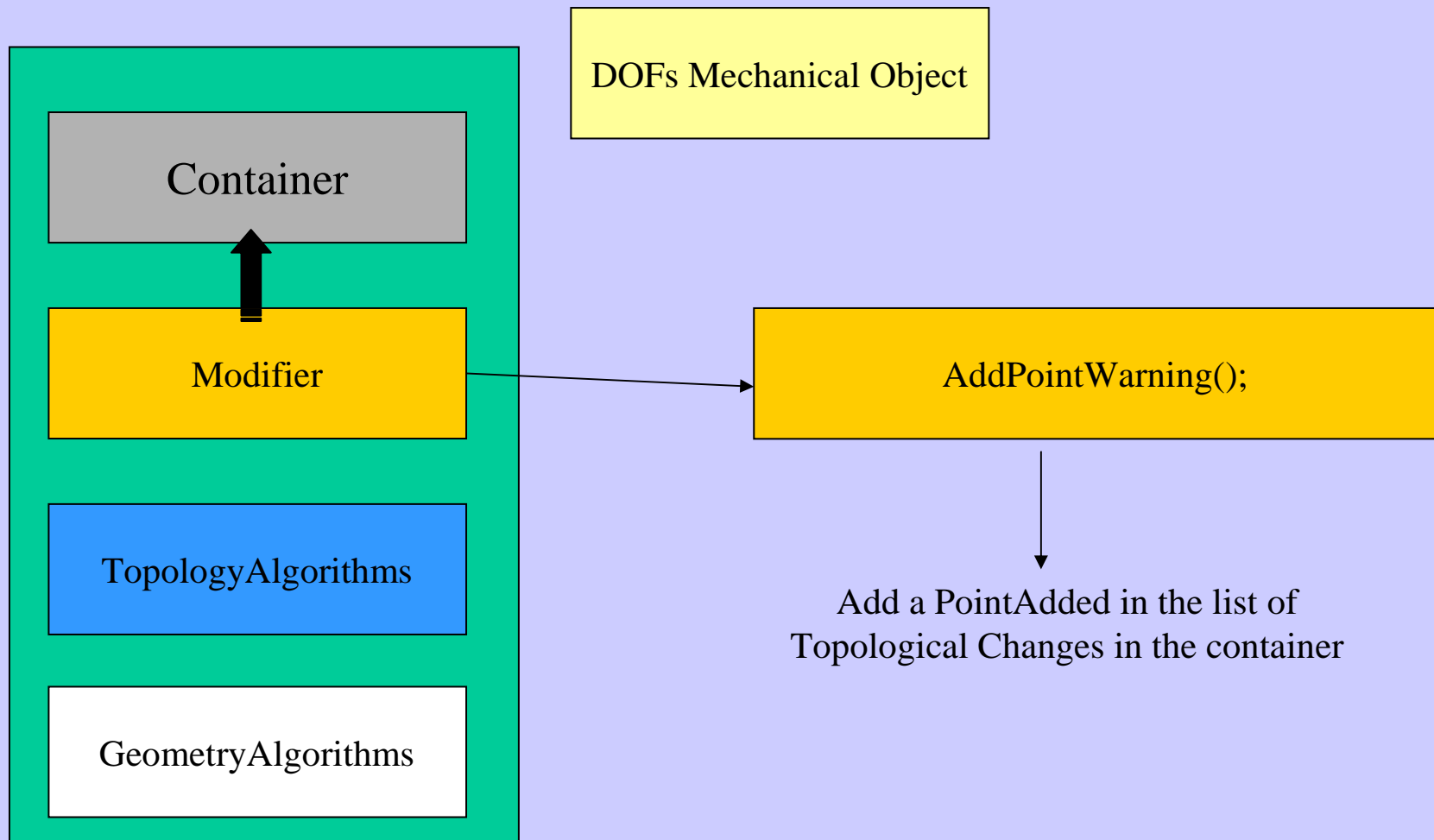
What happens when I split an Edge ?



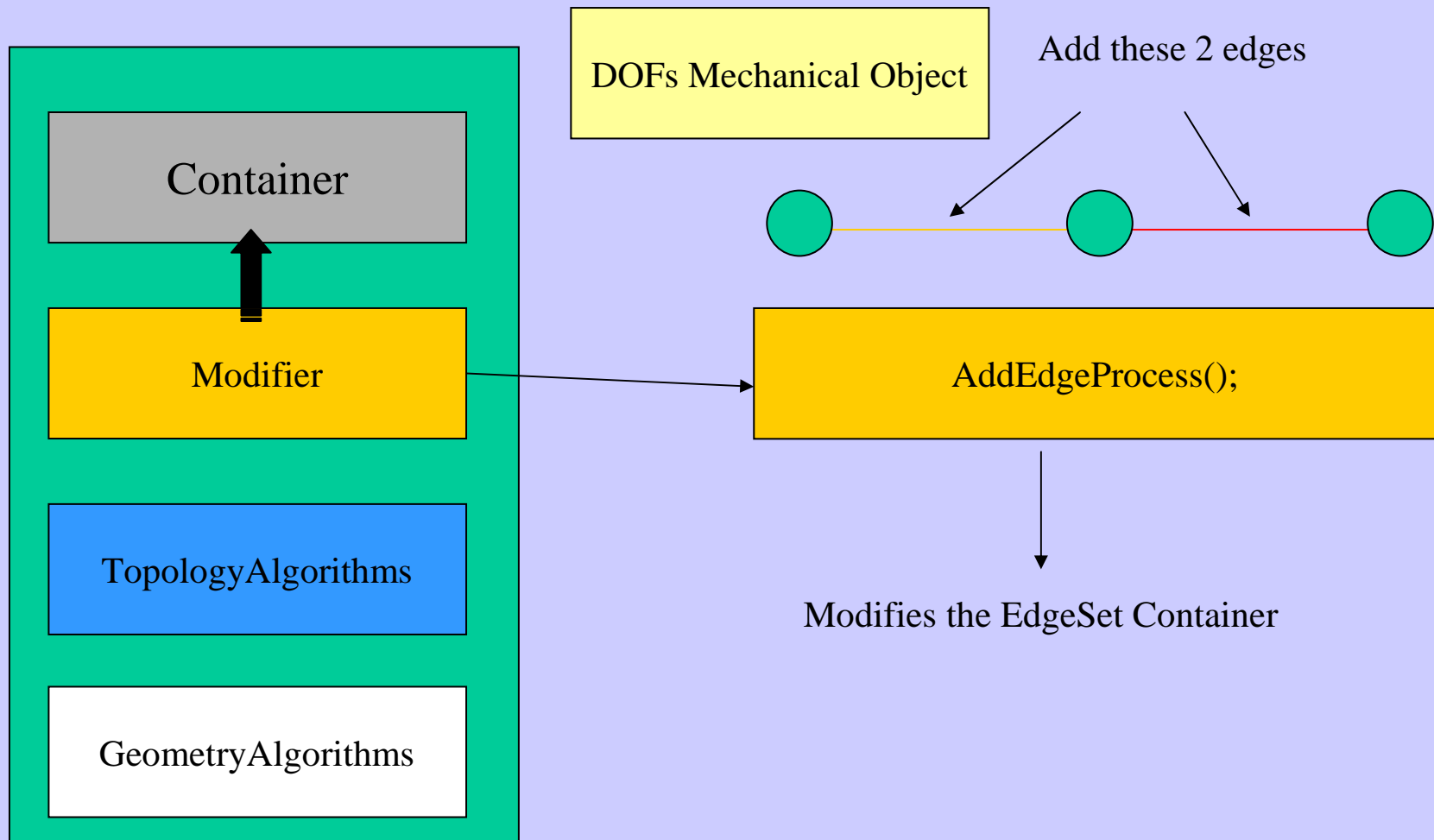
What happens when I split an Edge ?



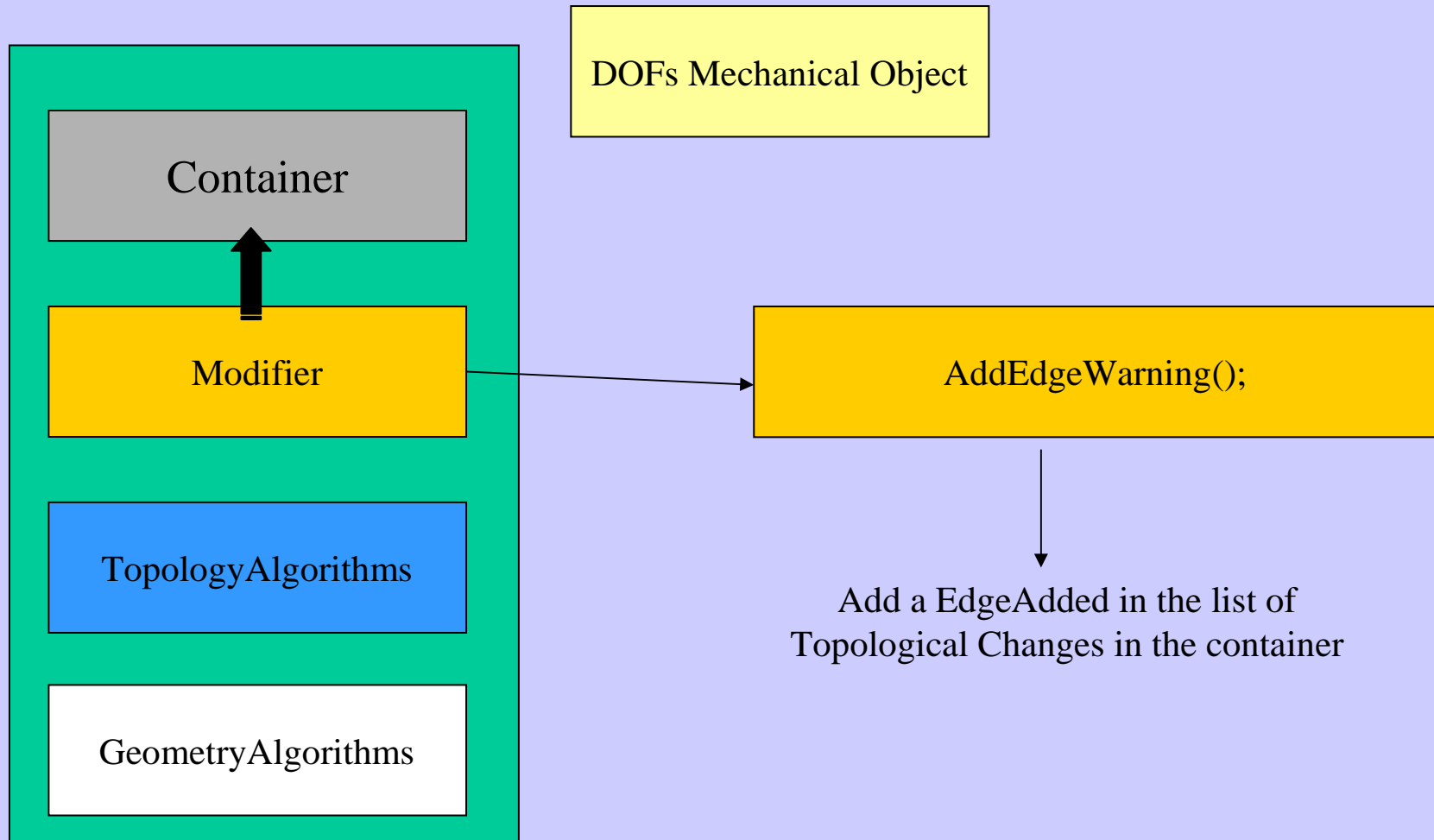
What happens when I split an Edge ?



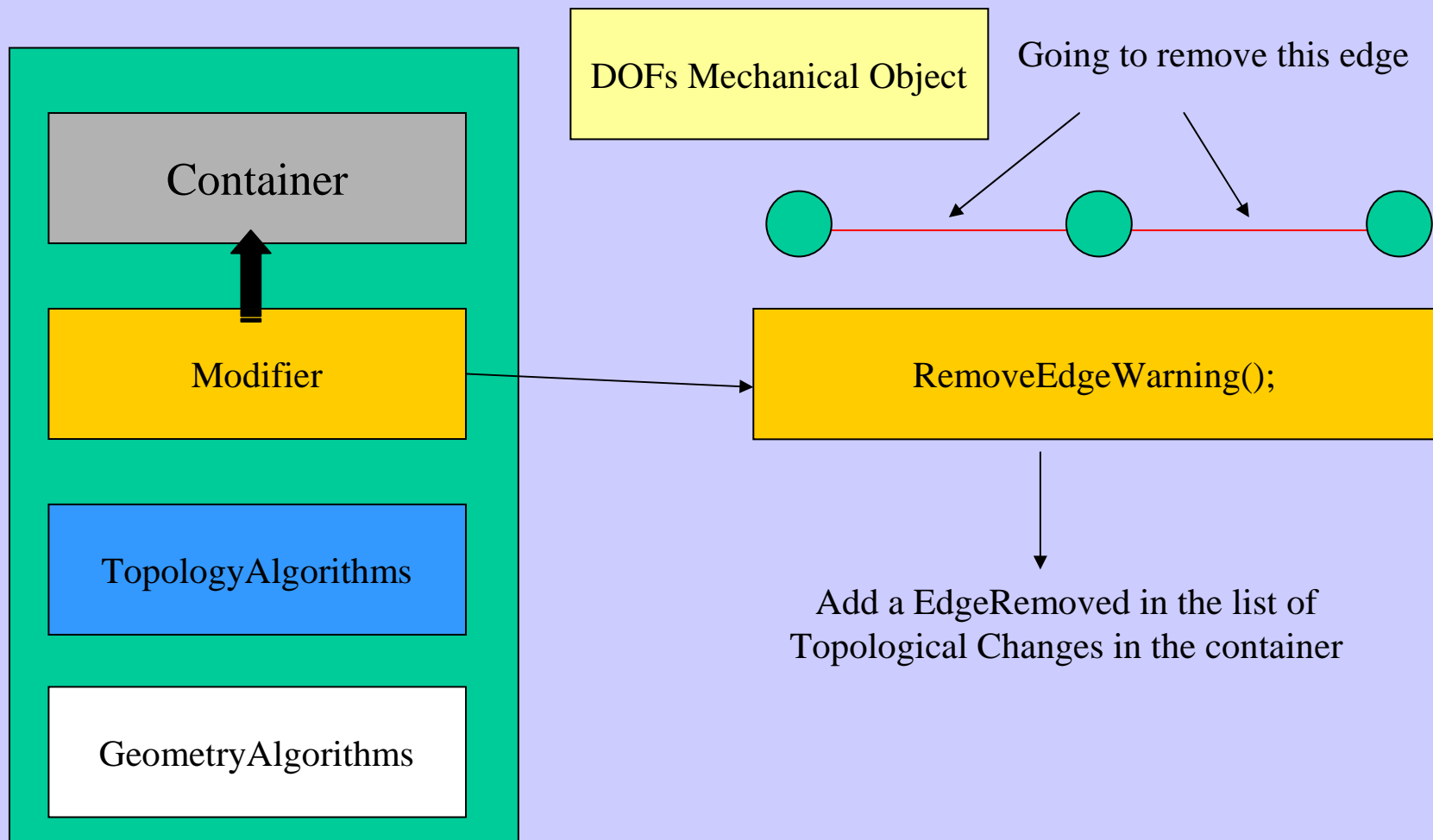
What happens when I split an Edge ?



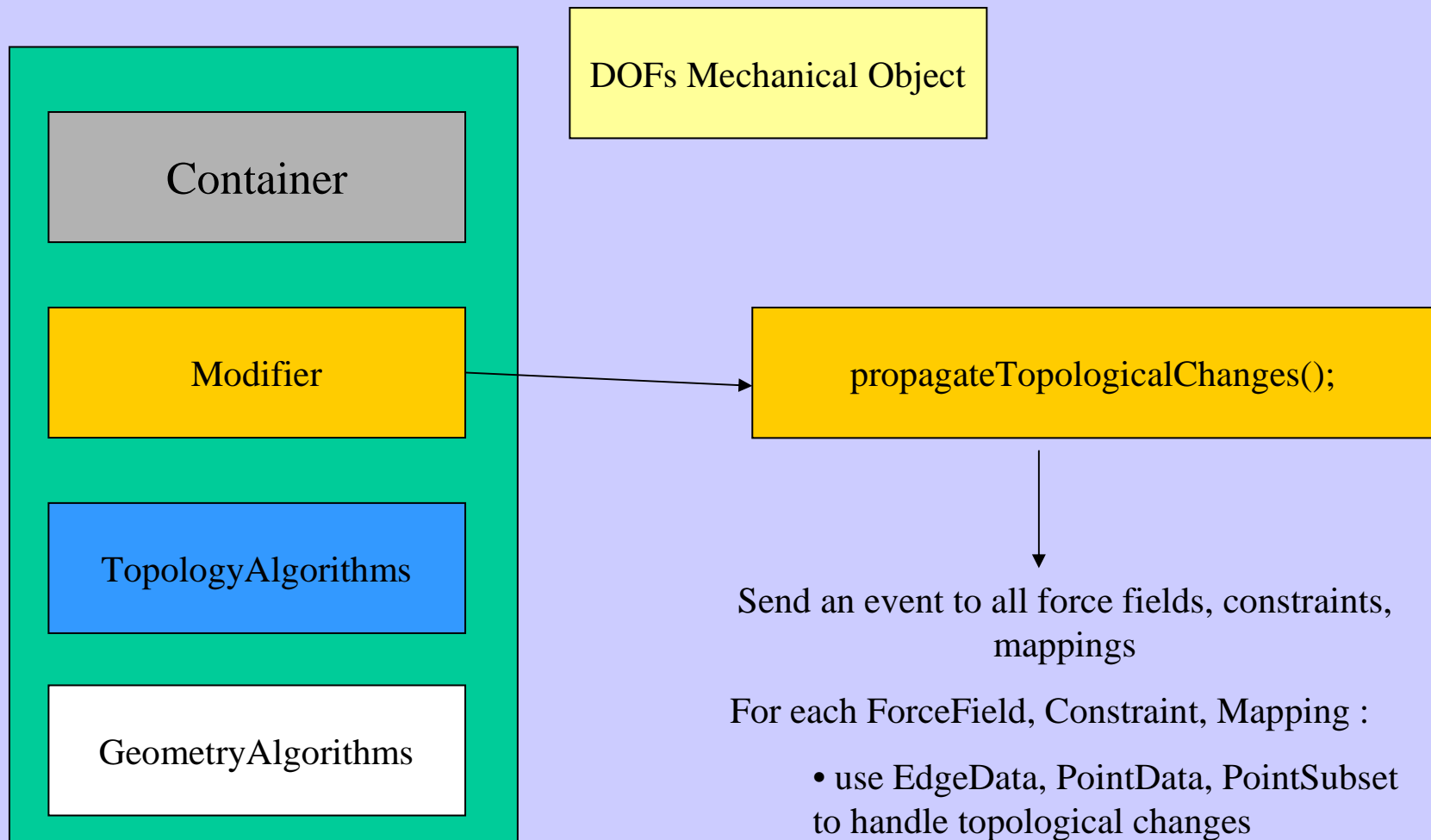
What happens when I split an Edge ?



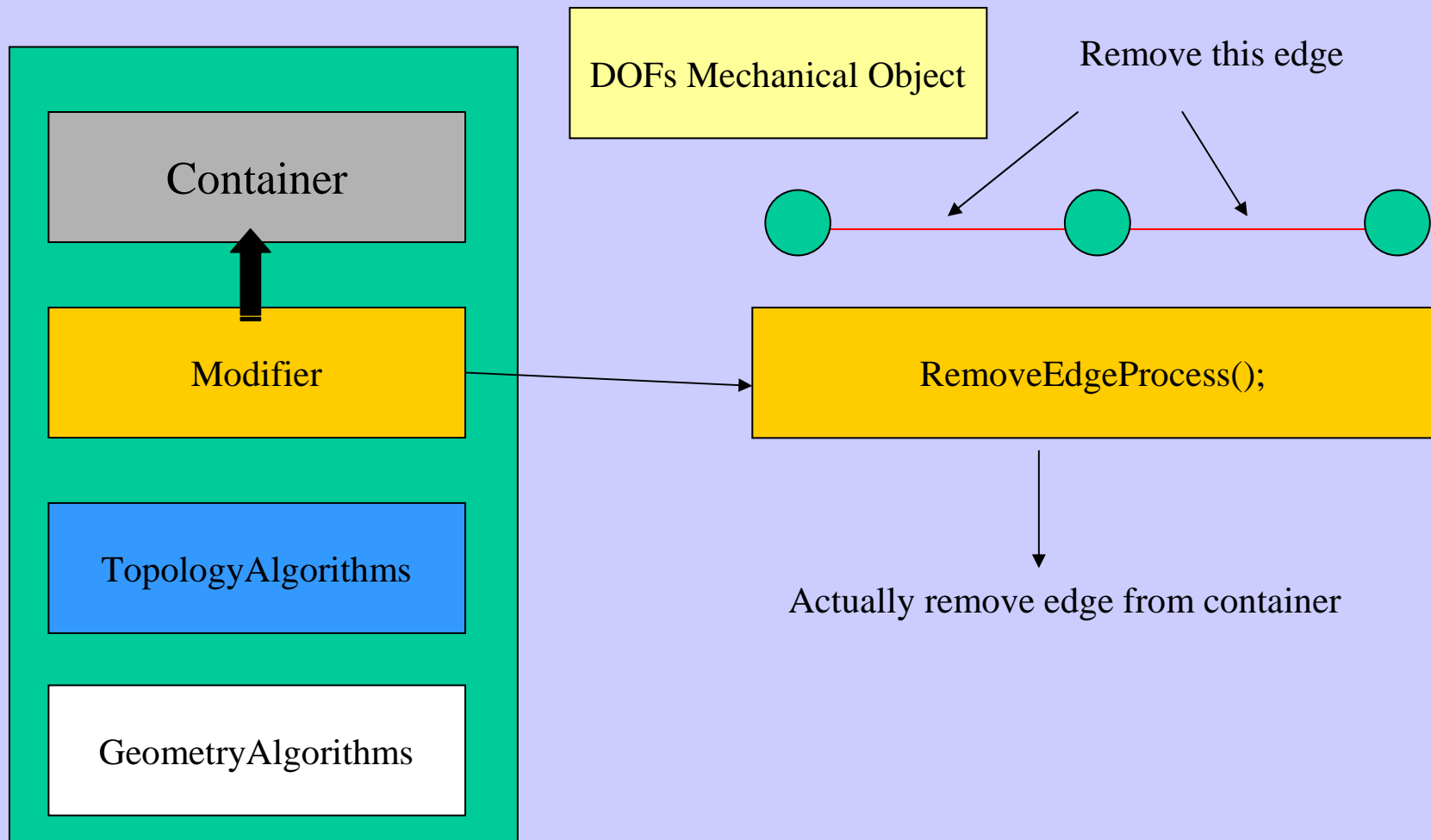
What happens when I split an Edge ?



What happens when I split an Edge ?

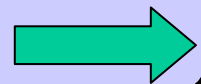


What happens when I split an Edge ?



Current Implementation of Mesh Topologies

- Only limited number of topologies implemented



Add new topology description, e.g. manifold triangulation or tetrahedralisation or regular grids

- Limited number of topology changes operations (removal of edge or triangle)



Add ability to cut triangle or tetrahedral meshes

- Must replace MeshTopology classes with basicTopology classes



Other issues

- Connection with computational geometry classes (like CGAL)
- Should the DOFs container be associated with the topology ?
- Handling of multiple topologies, topological mappings
- Definition of fields (temperature, potential...)



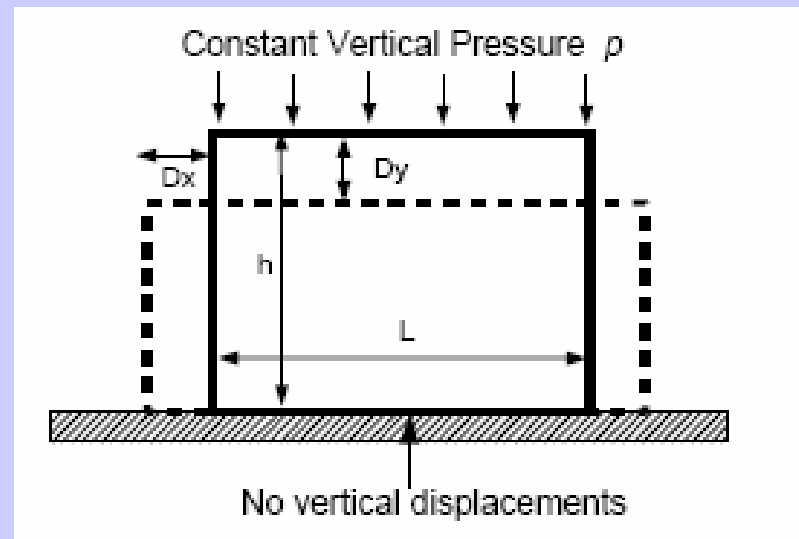
Benchmarking in SOFA

- SOFA is well-suited for benchmarking :
 - Internal Force computation
 - Collision detection algorithm
 - Solver
 - Haptics models
- Benchmark :
 - Computation times
 - Absolute or relative Errors



Very Preliminary attempt

- Benchmark Rectangular Membrane Deformation



- Knowledge of linear elastic solution ($Dy = \frac{pL^2}{8Eh}$, $Dx = \frac{pL}{2Eh}$)



Benchmarking triangle membrane

- Create a benchmark program that tests different material (Young Modulus and Poisson Ratio) with different force fields
- Benchmark program outputs ASCII file (.csv file)
- Use external program (excel) to post-process the benchmark data



Benchmarking triangle membrane

- How to add a new forcefield to the benchmark program ?

```
ForceField<MyTypes> *addTriangularLinearElastic(GNode *node, double youngModulus, double
poissonRatio)
{
    MechanicalObject<MyTypes>* square=dynamic_cast<MechanicalObject<MyTypes>*
>(node->getMechanicalModel());
    if (square) {
        TriangleFEMLinearForceField<MyTypes> *ff=new
TriangleFEMLinearForceField<MyTypes>(square);
        node->addObject(ff);
        ff->setYoungModulus(youngModulus);
        ff->setPoissonRatio(poissonRatio);
        return (ForceField<MyTypes> *)ff;
    }
    else return NULL;
}
```

Post-processing

- Example : Use Excel and VBA to chart the outcome.
- Many other possibilities



Future Work

- Propose benchmark programs for key components :
 - Elastic force on Tetrahedral, Quad, Hexahedron
 - Collision detection, solvers,...
 - Must find ways to parameterize components
- Dependence on hardware / memory / compiler
- Find a proper post-processing interface

