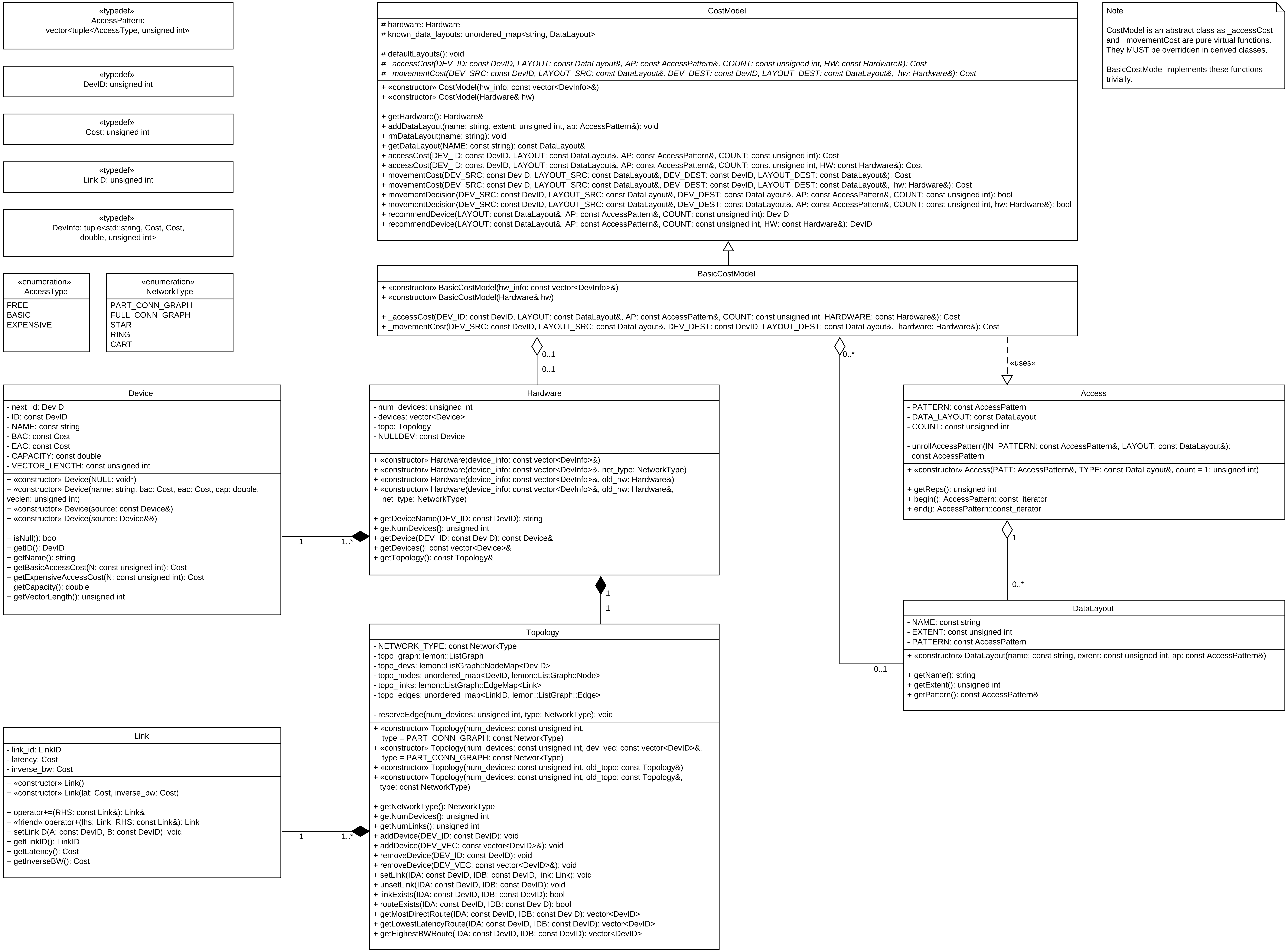

«typedef»
AccessPattern:
vector<tuple<AccessType, unsigned int»

«typedef»
DevID: unsigned int

«typedef»
Cost: unsigned int

«typedef»
LinkID: unsigned int

«typedef»
DevInfo: tuple<std::string, Cost, Cost,
double, unsigned int>

«enumeration»
AccessType

FREE
BASIC
EXPENSIVE

«enumeration»
NetworkType

PART_CONN_GRAPH
FULL_CONN_GRAPH
STAR
RING
CART

CostModel

# hardware: Hardware
# known_data_layouts: unordered_map<string, DataLayout>

# defaultLayouts(): void
# _accessCost(DEV_ID: const DevID, LAYOUT: const DataLayout&, AP: const AccessPattern&, COUNT: const unsigned int, HW: const Hardware&): Cost
# _movementCost(DEV_SRC: const DevID, LAYOUT_SRC: const DataLayout&, DEV_DEST: const DevID, LAYOUT_DEST: const DataLayout&, hw: Hardware&): Cost

+ «constructor» CostModel(hw_info: const vector<DevInfo>&)
+ «constructor» CostModel(Hardware& hw)

+ getHardware(): Hardware&
+ addDataLayout(name: string, extent: unsigned int, ap: AccessPattern&): void
+ rmDataLayout(name: string): void
+ getDataLayout(NAME: const string): const DataLayout&
+ accessCost(DEV_ID: const DevID, LAYOUT: const DataLayout&, AP: const AccessPattern&, COUNT: const unsigned int): Cost
+ accessCost(DEV_ID: const DevID, LAYOUT: const DataLayout&, AP: const AccessPattern&, COUNT: const unsigned int, HW: const Hardware&): Cost
+ movementCost(DEV_SRC: const DevID, LAYOUT_SRC: const DataLayout&, DEV_DEST: const DevID, LAYOUT_DEST: const DataLayout&): Cost
+ movementCost(DEV_SRC: const DevID, LAYOUT_SRC: const DataLayout&, DEV_DEST: const DevID, LAYOUT_DEST: const DataLayout&, hw: Hardware&): Cost
+ movementDecision(DEV_SRC: const DevID, LAYOUT_SRC: const DataLayout&, DEV_DEST: const DataLayout&, AP: const AccessPattern&, COUNT: const unsigned int): bool
+ movementDecision(DEV_SRC: const DevID, LAYOUT_SRC: const DataLayout&, DEV_DEST: const DataLayout&, AP: const AccessPattern&, COUNT: const unsigned int, hw: Hardware&): bool
+ recommendDevice(LAYOUT: const DataLayout&, AP: const AccessPattern&, COUNT: const unsigned int): DevID
+ recommendDevice(LAYOUT: const DataLayout&, AP: const AccessPattern&, COUNT: const unsigned int, HW: const Hardware&): DevID

Note

CostModel is an abstract class as _accessCost
and _movementCost are pure virtual functions.
They MUST be overridden in derived classes.

BasicCostModel implements these functions
trivially.

BasicCostModel

+ «constructor» BasicCostModel(hw_info: const vector<DevInfo>&)
+ «constructor» BasicCostModel(Hardware& hw)

+ _accessCost(DEV_ID: const DevID, LAYOUT: const DataLayout&, AP: const AccessPattern&, COUNT: const unsigned int, HARDWARE: const Hardware&): Cost
+ _movementCost(DEV_SRC: const DevID, LAYOUT_SRC: const DataLayout&, DEV_DEST: const DevID, LAYOUT_DEST: const DataLayout&, hardware: Hardware&): Cost

Device

- next_id: DevID
- ID: const DevID
- NAME: const string
- BAC: const Cost
- EAC: const Cost
- CAPACITY: const double
- VECTOR_LENGTH: const unsigned int

+ «constructor» Device(NULL: void*)
+ «constructor» Device(name: string, bac: Cost, eac: Cost, cap: double,
veclen: unsigned int)
+ «constructor» Device(source: const Device&)
+ «constructor» Device(source: Device&&)

+ isNull(): bool
+ getID(): DevID
+ getName(): string
+ getBasicAccessCost(N: const unsigned int): Cost
+ getExpensiveAccessCost(N: const unsigned int): Cost
+ getCapacity(): double
+ getVectorLength(): unsigned int

Hardware

- num_devices: unsigned int
- devices: vector<Device>
- topo: Topology
- NULLDEV: const Device

+ «constructor» Hardware(device_info: const vector<DevInfo>&)
+ «constructor» Hardware(device_info: const vector<DevInfo>&, net_type: NetworkType)
+ «constructor» Hardware(device_info: const vector<DevInfo>&, old_hw: Hardware&)
+ «constructor» Hardware(device_info: const vector<DevInfo>&, old_hw: Hardware&,
net_type: NetworkType)

+ getDeviceName(DEV_ID: const DevID): string
+ getNumDevices(): unsigned int
+ getDevice(DEV_ID: const DevID): const Device&
+ getDevices(): const vector<Device>&
+ getTopology(): const Topology&

Access

- PATTERN: const AccessPattern
- DATA_LAYOUT: const DataLayout
- COUNT: const unsigned int

- unrollAccessPattern(IN_PATTERN: const AccessPattern&, LAYOUT: const DataLayout&):
const AccessPattern

+ «constructor» Access(PATT: AccessPattern&, TYPE: const DataLayout&, count = 1: unsigned int)

+ getReps(): unsigned int
+ begin(): AccessPattern::const_iterator
+ end(): AccessPattern::const_iterator

DataLayout

- NAME: const string
- EXTENT: const unsigned int
- PATTERN: const AccessPattern

+ «constructor» DataLayout(name: const string, extent: const unsigned int, ap: const AccessPattern&)

+ getName(): string
+ getExtent(): unsigned int
+ getPattern(): const AccessPattern&

Topology

- NETWORK_TYPE: const NetworkType
- topo_graph: lemon::ListGraph
- topo_devs: lemon::ListGraph::NodeMap<DevID>
- topo_nodes: unordered_map<DevID, lemon::ListGraph::Node>
- topo_links: lemon::ListGraph::EdgeMap<Link>
- topo_edges: unordered_map<LinkID, lemon::ListGraph::Edge>

- reserveEdge(num_devices: unsigned int, type: NetworkType): void

+ «constructor» Topology(num_devices: const unsigned int,
type = PART_CONN_GRAPH: const NetworkType)
+ «constructor» Topology(num_devices: const unsigned int, dev_vec: const vector<DevID>&,
type = PART_CONN_GRAPH: const NetworkType)
+ «constructor» Topology(num_devices: const unsigned int, old_topo: const Topology&)
+ «constructor» Topology(num_devices: const unsigned int, old_topo: const Topology&,
type: const NetworkType)

+ getNetworkType(): NetworkType
+ getNumDevices(): unsigned int
+ getNumLinks(): unsigned int
+ addDevice(DEV_ID: const DevID): void
+ addDevice(DEV_VEC: const vector<DevID>&): void
+ removeDevice(DEV_ID: const DevID): void
+ removeDevice(DEV_VEC: const vector<DevID>&): void
+ setLink(IDA: const DevID, IDB: const DevID, link: Link): void
+ unsetLink(IDA: const DevID, IDB: const DevID): void
+ linkExists(IDA: const DevID, IDB: const DevID): bool
+ routeExists(IDA: const DevID, IDB: const DevID): bool
+ getMostDirectRoute(IDA: const DevID, IDB: const DevID): vector<DevID>
+ getLowestLatencyRoute(IDA: const DevID, IDB: const DevID): vector<DevID>
+ getHighestBWRoute(IDA: const DevID, IDB: const DevID): vector<DevID>

Link

- link_id: LinkID
- latency: Cost
- inverse_bw: Cost

+ «constructor» Link()
+ «constructor» Link(lat: Cost, inverse_bw: Cost)

+ operator+=(RHS: const Link&): Link&
+ «friend» operator+(lhs: Link, RHS: const Link&): Link
+ setLinkID(A: const DevID, B: const DevID): void
+ getLinkID(): LinkID
+ getLatency(): Cost
+ getInverseBW(): Cost