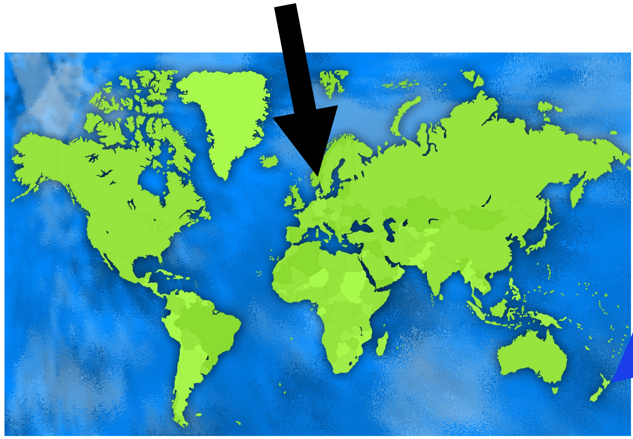
UiO : Department of Informatics
University of Oslo

DCEP-Sim: An Open Simulation Framework for Distributed CEP

Introduction for Users and Prospective Developers

Fabrice Starks

Stein Kristiansen, Thomas Plagemann



Introduction and Motivation

- Data streams and information flow processing
  - Financial tickers
  - Traffic management
  - Internet of Things
  - eHealth
- Real-time processing:
  - Data Stream Management Systems
  - Complex Event Processing

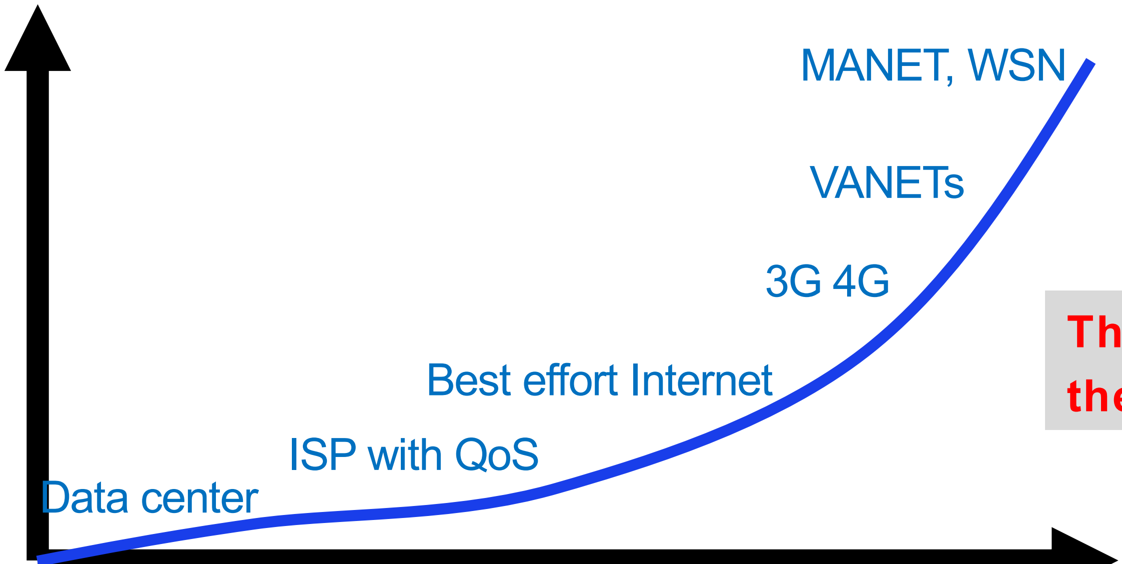
Distributed CEP

- CEP instances communicate via a network
  - End to end delay
  - Error rate
  - Available bandwidth
- How deterministic are the network properties
  - Guaranteed QoS vs. best effort
  - Private vs. public networks
  - Static vs. mobile networks

[Source: https://www.pcsteps.com/10751-mobile-internet-e-3g-h-plus-4g-mobile-network/]

Distributed CEP - Challenges

- Test and evaluate
- Real world vs. emulation vs. simulation
- What are *realistic, representative* network properties?



The more dynamics, the harder

Some insights from a recent survey

- Starks, F., Plagemann, T., Goebel, V., Kristiansen, S. (2018). **Mobile Distributed Complex Event Processing - Ubi Sumus? Quo Vadimus?**, In *Mobile Big Data: A Roadmap from Models to Technologies*. Springer
- 13 publications on mobile Distributed CEP with 19 evaluation reports
  - 2 based on mathematical modeling
  - 3 based on PlanetLab experiments
  - 3 based on emulation
  - 11 based on simulation
    - 7 based on simulators created for the specific experiments
    - 4 based on popular network simulators (J-Sim, OMNet and PeerSim)

- The missing consensus on evaluation approaches motivated our development of DCEP-Sim (presented at DEBS 2017)

Aim of this tutorial

- For us:
  - Start an open source project with DCEP-Sim

- For you (assuming 3 types of attendees):
  - Explain what you could do with DCEP-Sim in your work
  - How to get started with DCEP-Sim
  - How to use DCEP-Sim in your research and contribute to the code base

Disclaimer

- DCEP-Sim is
  - not a commercial product,
  - but an outcome of the PhD thesis from Fabrice Starks
  - and is now open to contributions from the community

- DCEP-Sim inherits strength and weaknesses of ns-3
  - many high quality network models
  - high flexibility
  - powerful tracing and data collection
  - efficient
  - software execution time is not considered

Outline

- Introduction and motivation
- Concepts and architecture of the distributed CEP engine in DCEP-Sim
  - Requirements
  - Design principles
  - CEP engine
  - Placement
  - Overall architecture
- Introduction to the network simulator ns-3
  - Principles of discrete event simulation
  - ns-3 Overview
  - Key ns-3 modeling and simulation concepts
  - Fundamental ns-3 models
  - ns-3 simulation via example

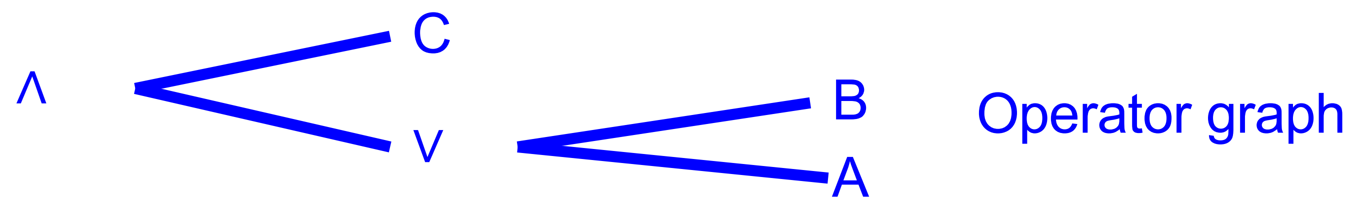
Outline (cont.)

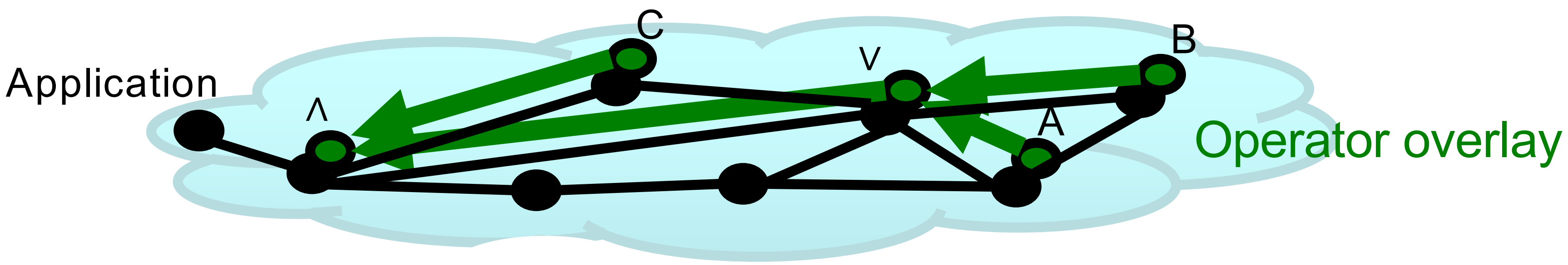
- DCEP-Sim use and extensions
  - Overview code structure
  - How do I run DCEP-Sim & how works a «script»
  - Changing the workload
  - How are placement policies implemented -> adding new placement
  - How are operators implemented -> adding new operators
- Conclusions

- Hands-on if you want to install ns-3 and run DCEP-Sim on you Linux laptop

Placement the Main Challenge of Distributed CEP

Query: (A ∨ B) ∧ C





Network node   Event Broker

Placement the Main Challenge of Distributed CEP

Where to place the operators?

Network link properties & overlay link properties:

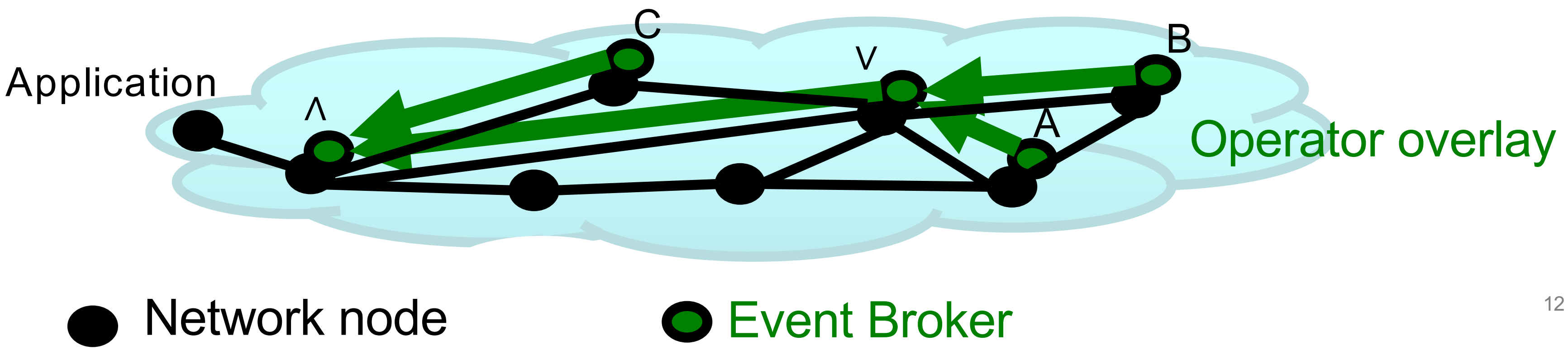
Latency, available bandwidth, loss

Traffic properties:

High event rate vs. low event rate from sources

Selectivity of operators

Other concerns: node resources, constraints, security

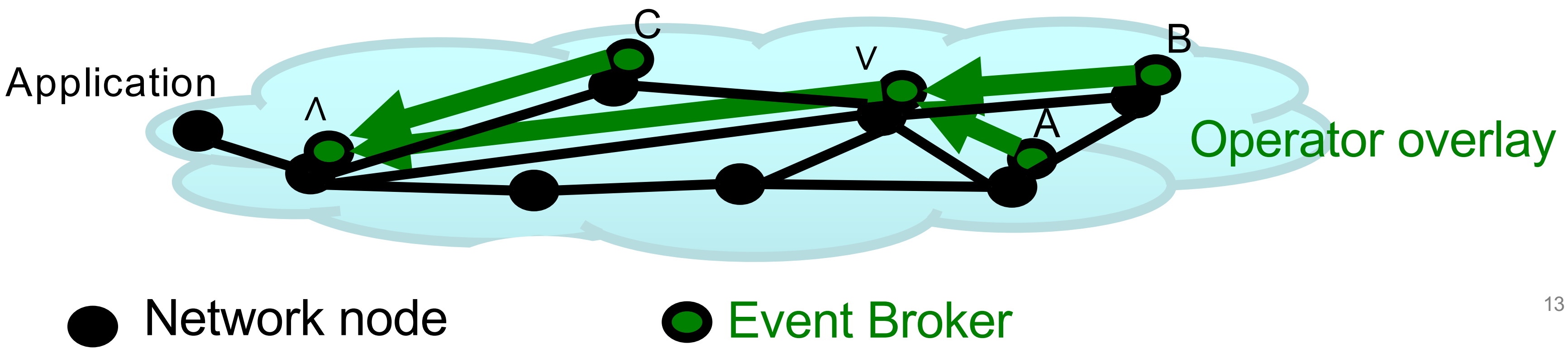


Placement the Main Challenge of Distributed CEP

What do you do if you have some cool new ideas for placement?

Model, design, implement

Test & implement – but how? → **DCEPSim**



DCEP-Sim Goals

- Tool for experimentation with Distributed CEP solutions
- Realistic models of various network types and conditions
- Ability to create arbitrary traffic patterns
- Support CEP query and query processing concepts
  - Operators, windows, selection policy, consumption policy
  - without the need to implement a »full CQL»
- Extensibility and flexibility
- Easy to use

Major Design Decisions

- Use the well established network simulator ns-3
  - Benefit from many years effort
  - Many existing models for link, network, transport level protocols, ++
  - High degree of realism
  - Tools for debugging, tracing, data collection, ++
- Simulation instead of emulation
  - Scalability

Engineering Principles

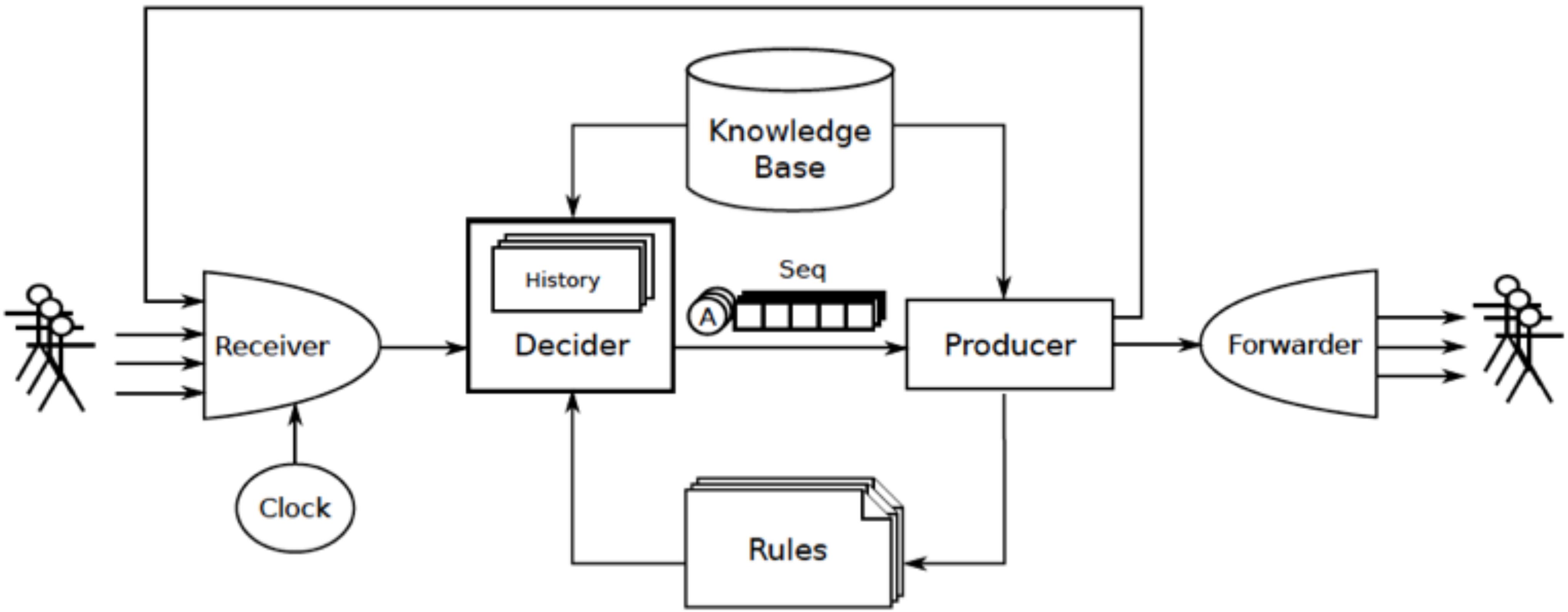
- Separation of concerns

- Separation of mechanisms and policies

Design & Implementation Approach

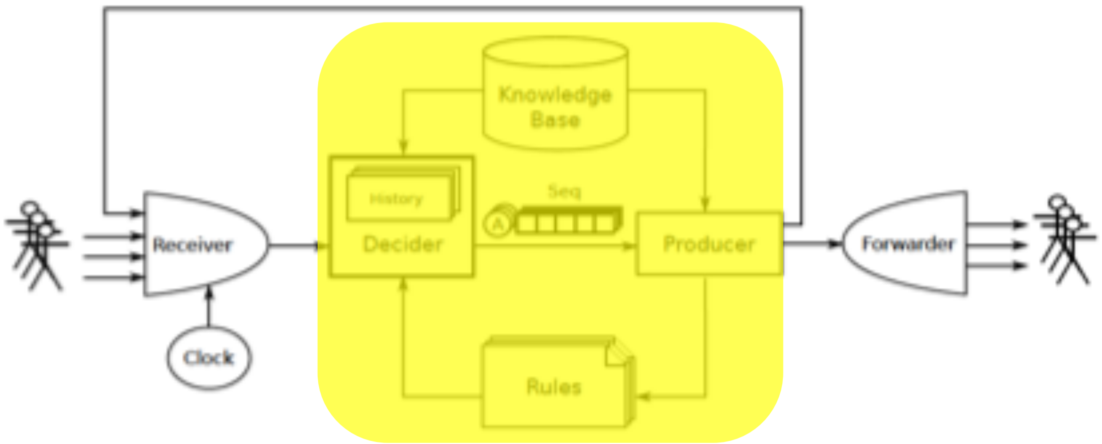
- Start:
  - Gianpaolo Cugola and Alessandro Margara. 2012. *Processing Flows of Information: From Data Stream to Complex Event Processing*. ACM Computing Surveys 44, June 2012
- Apply the engineering principles to develop the architecture
- Components & sub-components are good candidates to be implemented as objects
- Leverage the ns-3 features for the implementation of an extensible and flexible solution

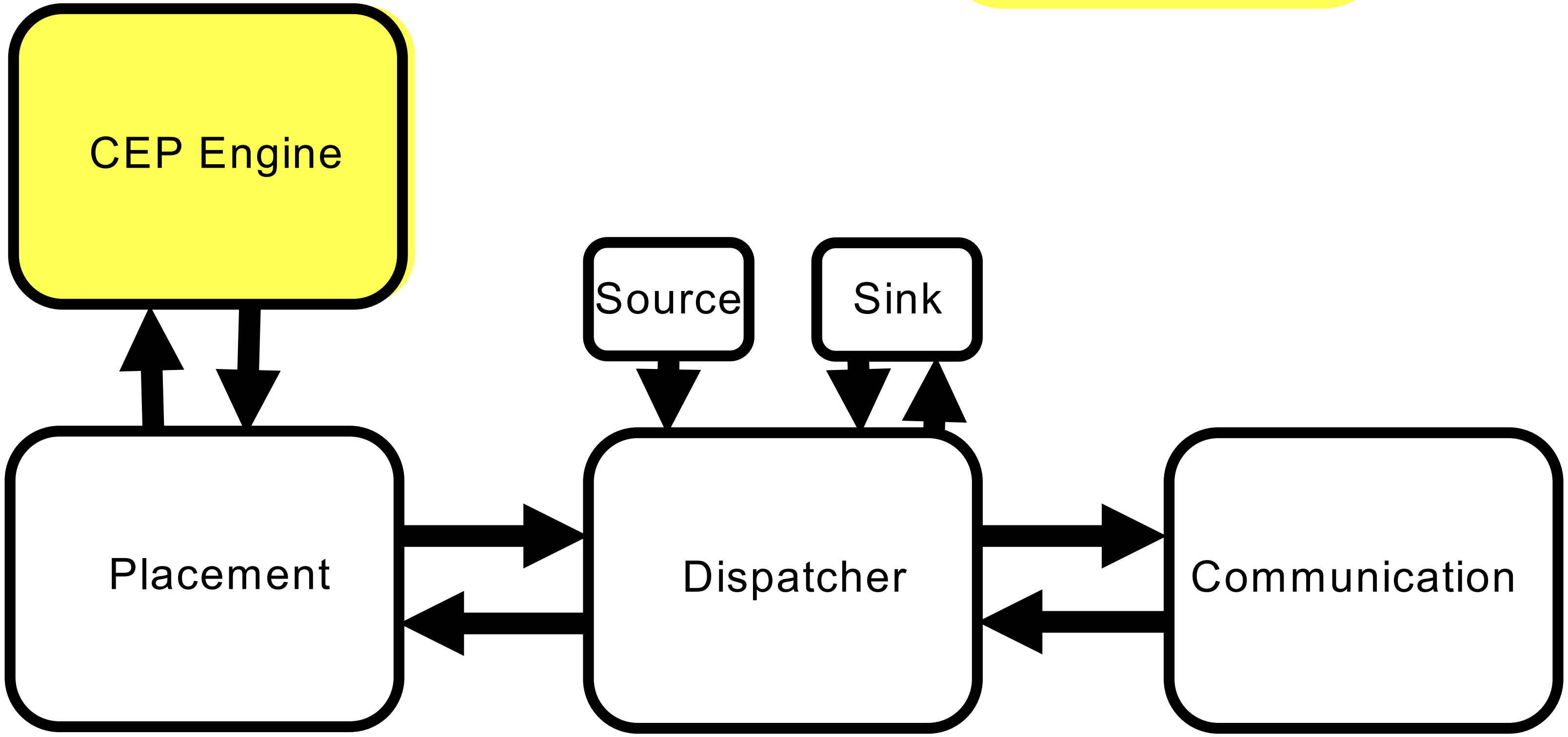
Functional Architecture of an IFP System



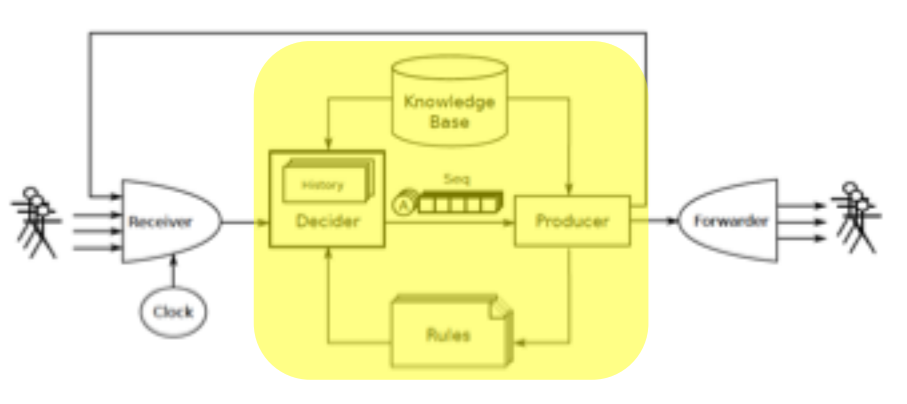
[Cugola et al. 2012]

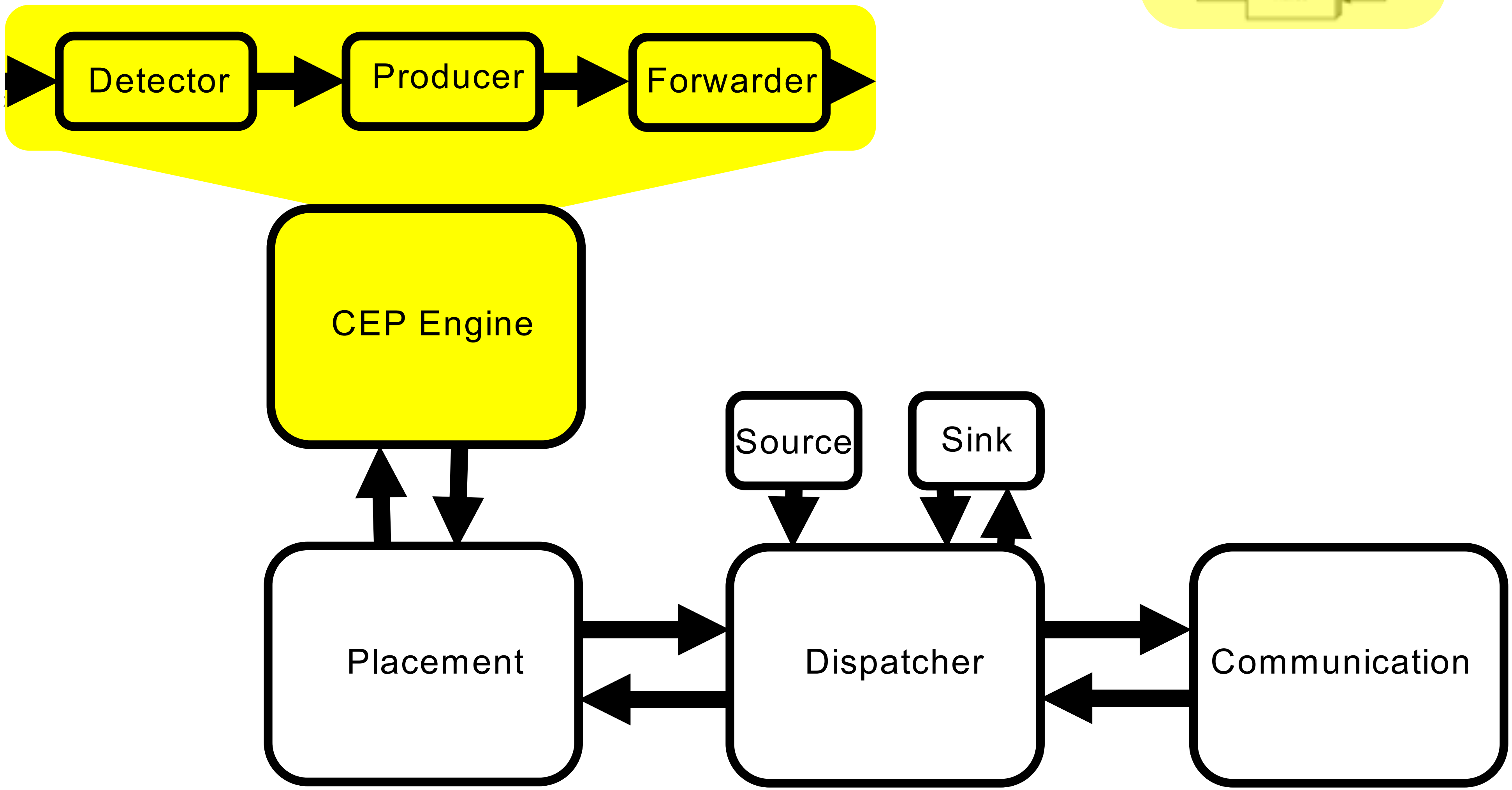
DCEP-Sim Components



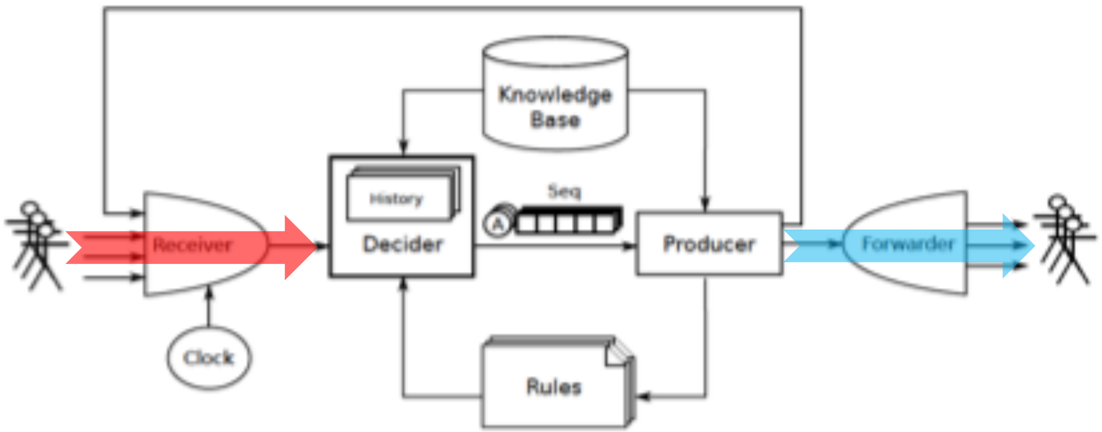


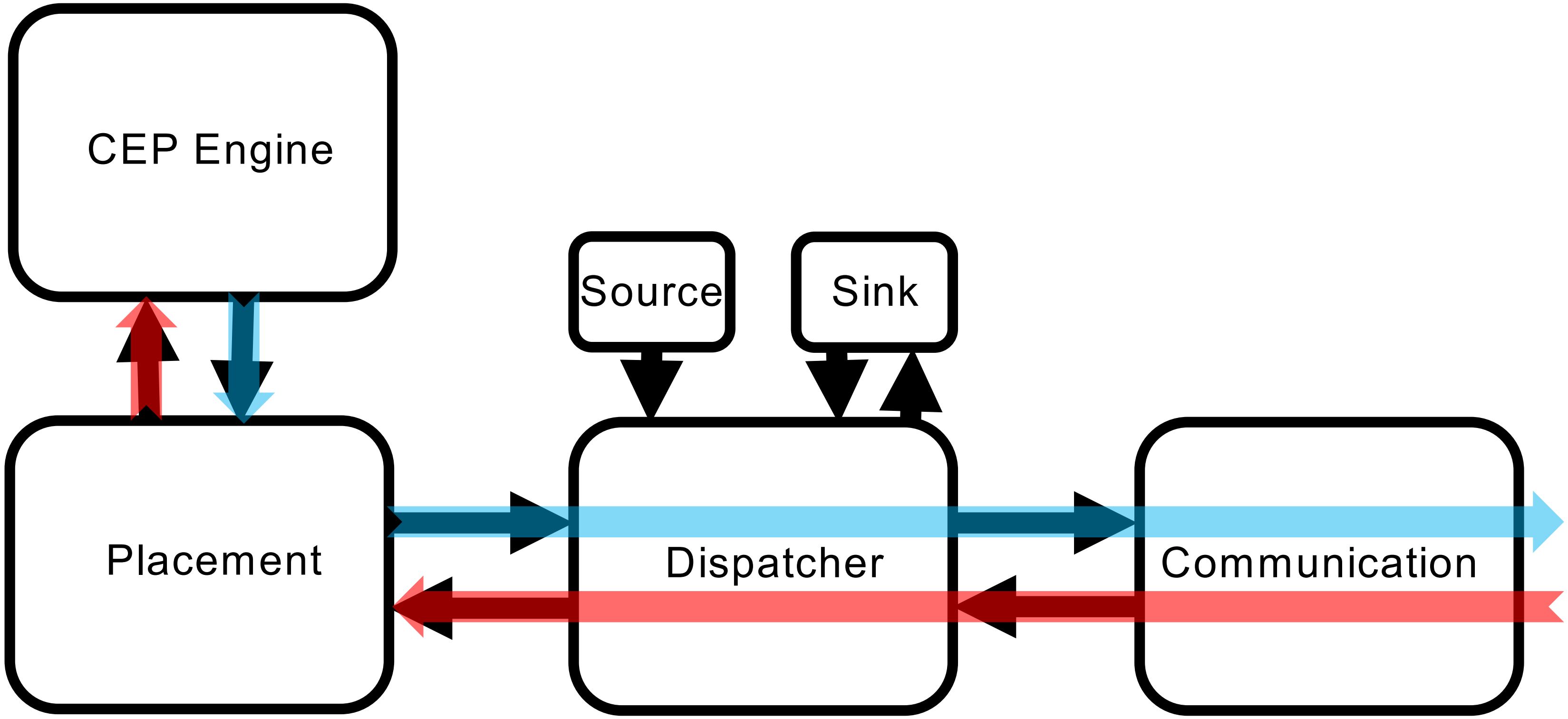
CEP-Engine


Knowledge Base
Decider
Seq
Producer
Receiver
Forwarder
Clock
Rules

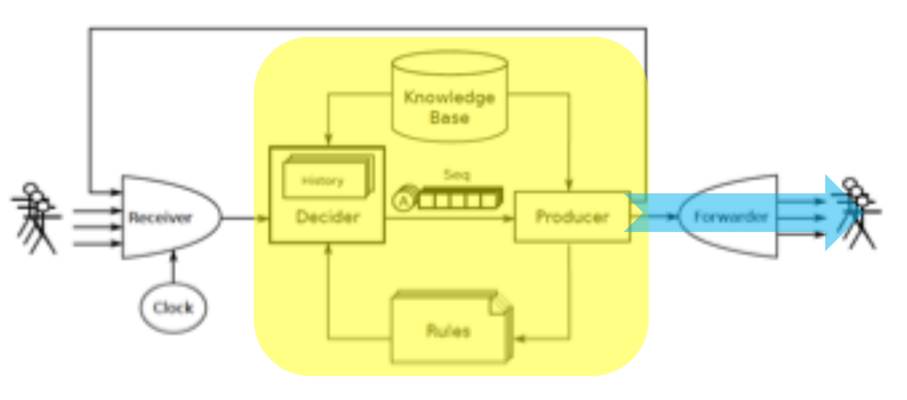

Detector
Producer
Forwarder
CEP Engine
Source
Sink
Placement
Dispatcher
Communication

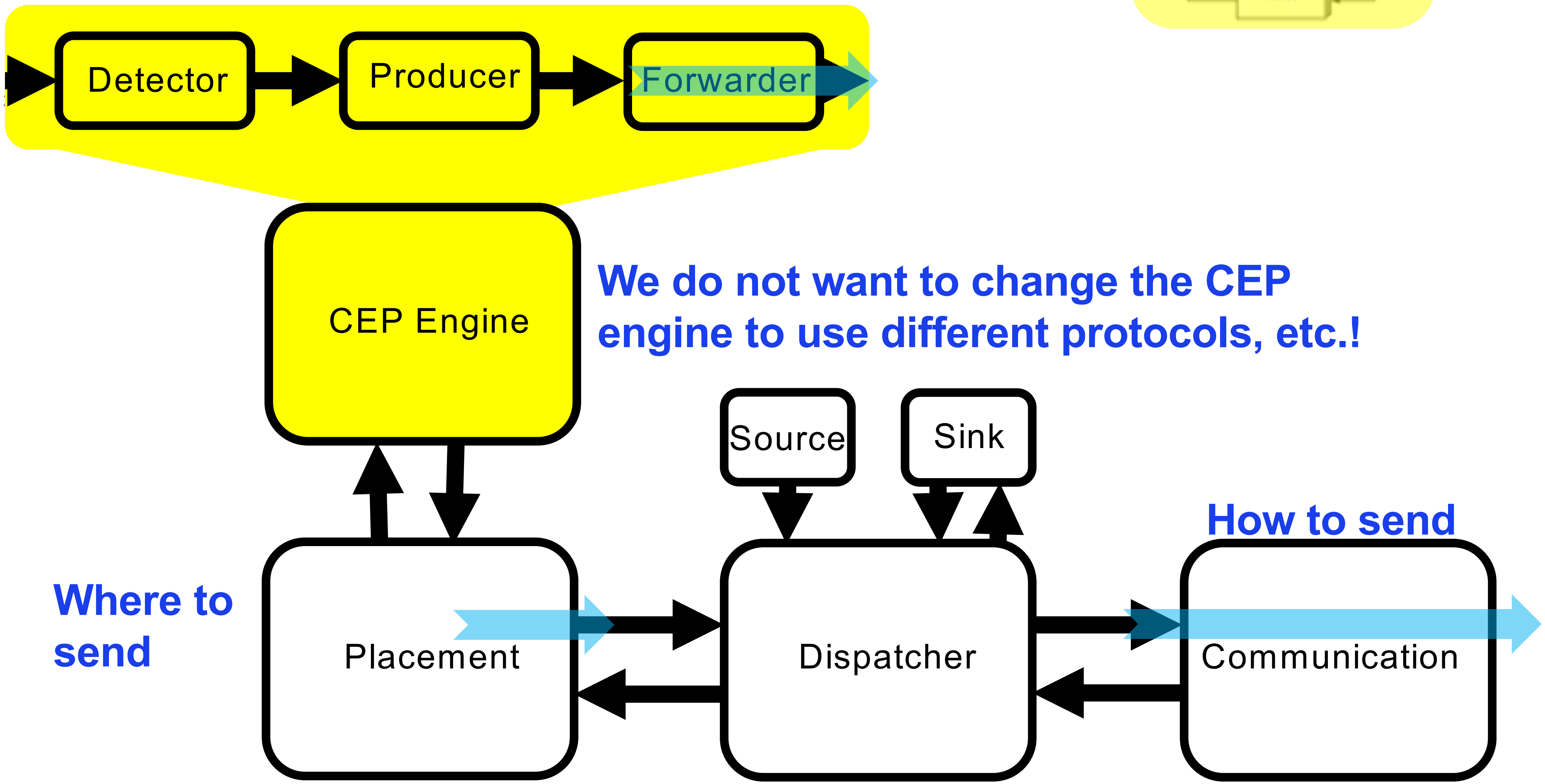
DCEP-Sim Components





Forwarder vs. Communication



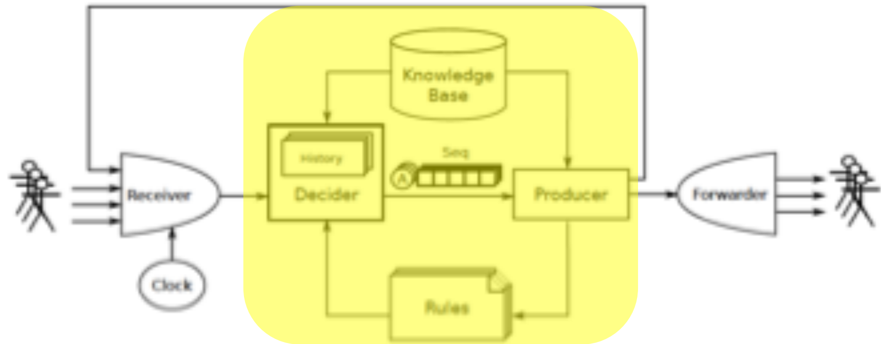


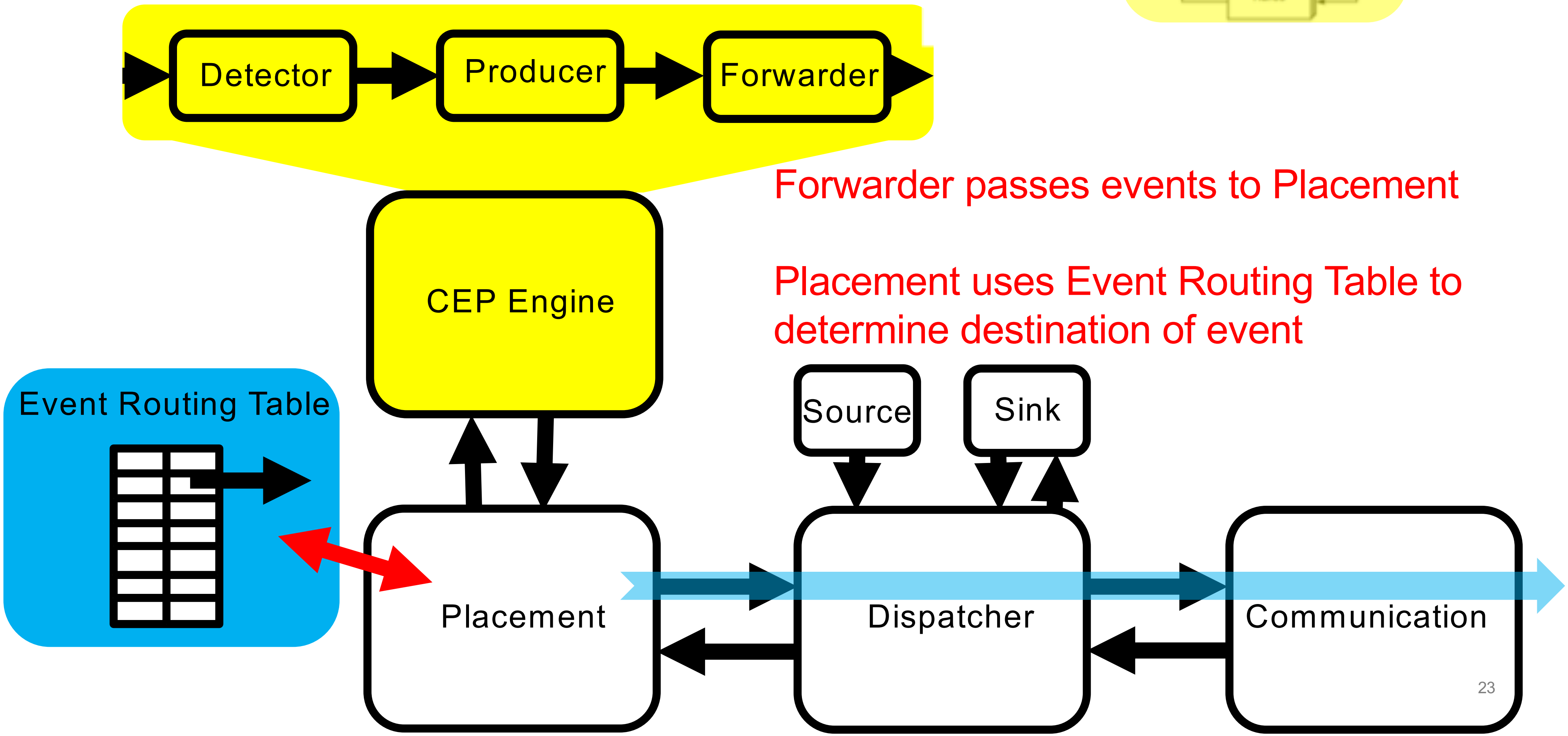
We do not want to change the CEP engine to use different protocols, etc.!

How to send

Where to send

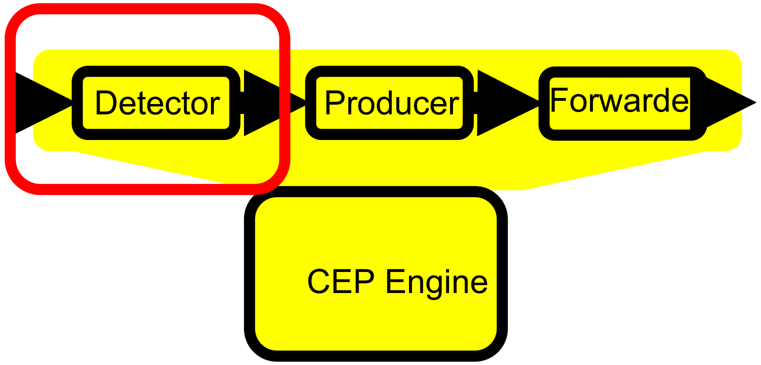
Forwarder & Placement



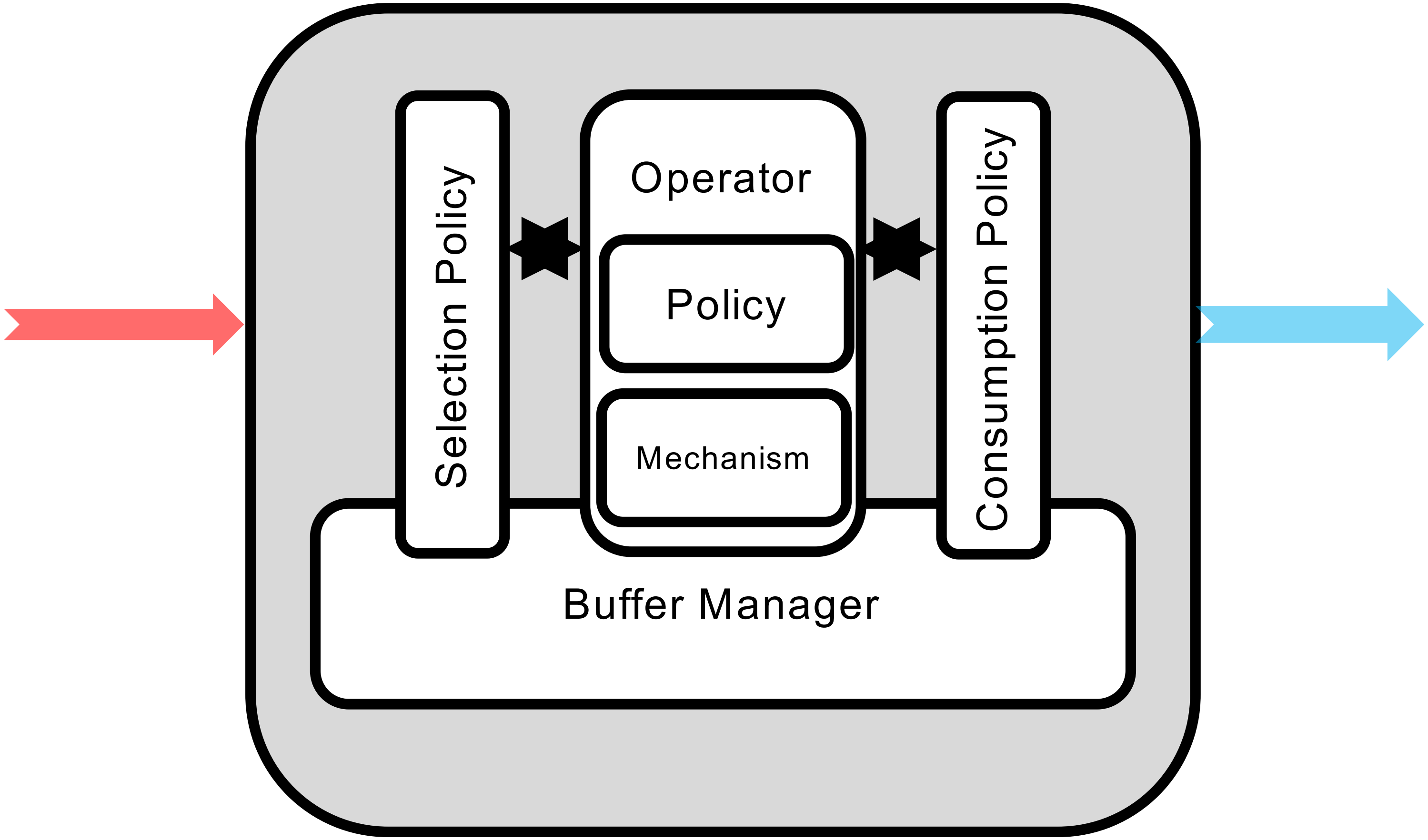


Forwarder passes events to Placement

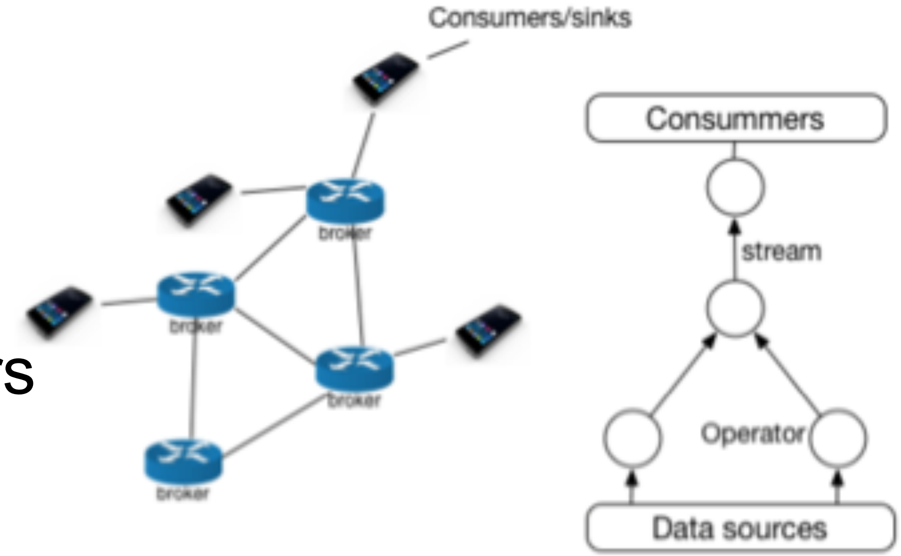
Placement uses Event Routing Table to determine destination of event



Operator in Detector



Placement

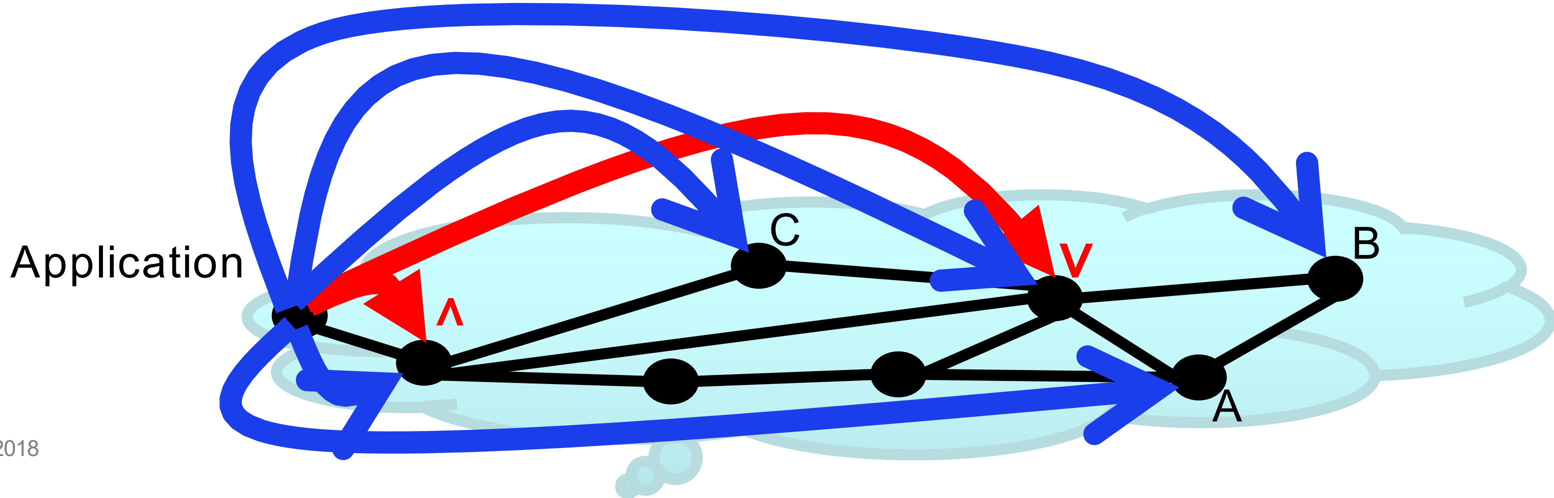


[Koldehofe et al. 2012]

- Assign operators to event brokers
  - Initial
  - Adaptation
  - Challenging optimization problems
    - Network utilization
    - Energy consumption
    - Event delivery latency
    - (security) constraints
- Result of placement: Operator overlay resp. operator tree
- Further tasks: event routing & forwarding

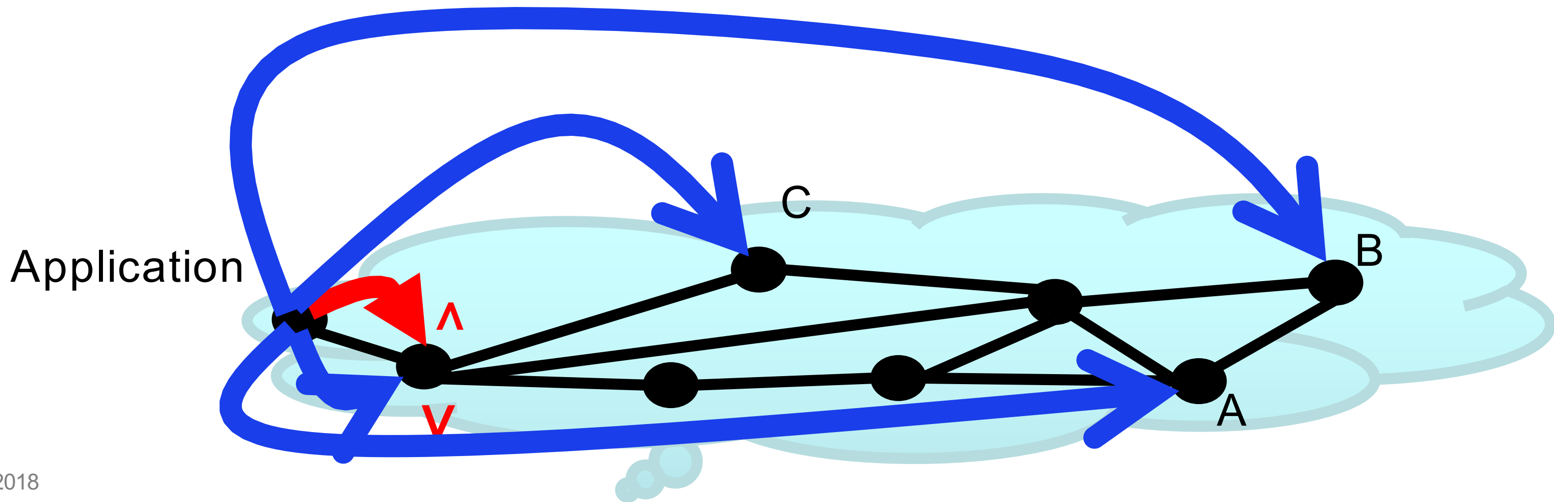
Example: Centralized Placement

- Sink node knows network topology
- Could calculate optimal placement for (A ∨ B) ∧ C
- Sends the operators to the selected brokers
- Sends routing information to all overlay nodes



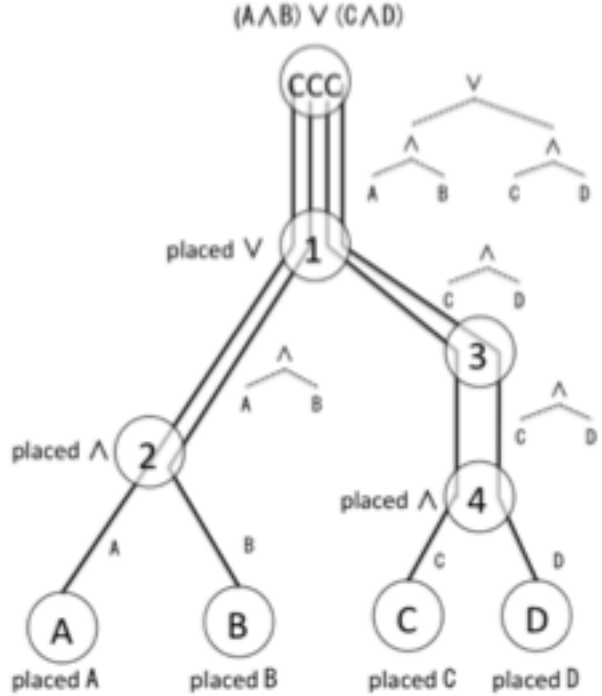
Example: Centralized Placement as it is in the Code

- Places the entire query on one node
- Sends the operators to the selected broker
- Sends routing information to all overlay nodes



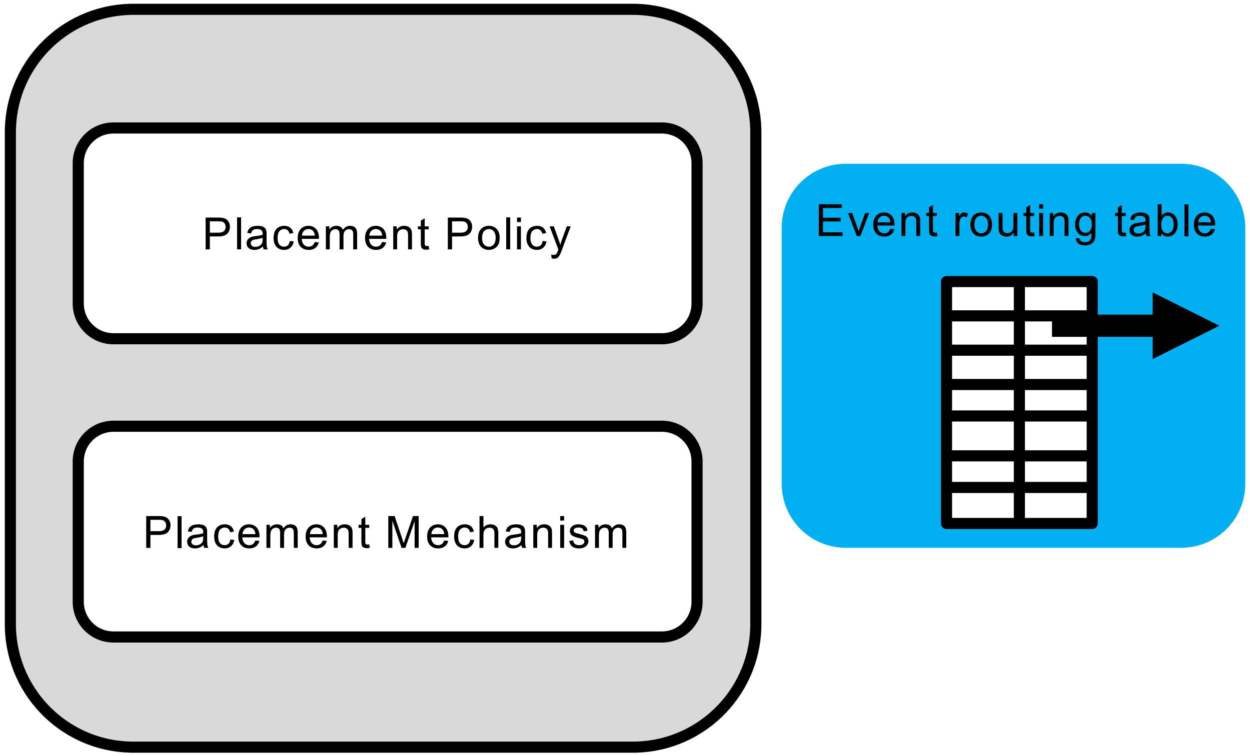
Example: Distributed Placement

- Sink (CCC) forwards operator graph on the shortes path towards sources
- On each following node:
  - can all sources reached through a single link?
    - Yes: forward entire (sub-)graph
    - No: split operator graph, place operator
      locally forward sub-graphs,
      update event routing table



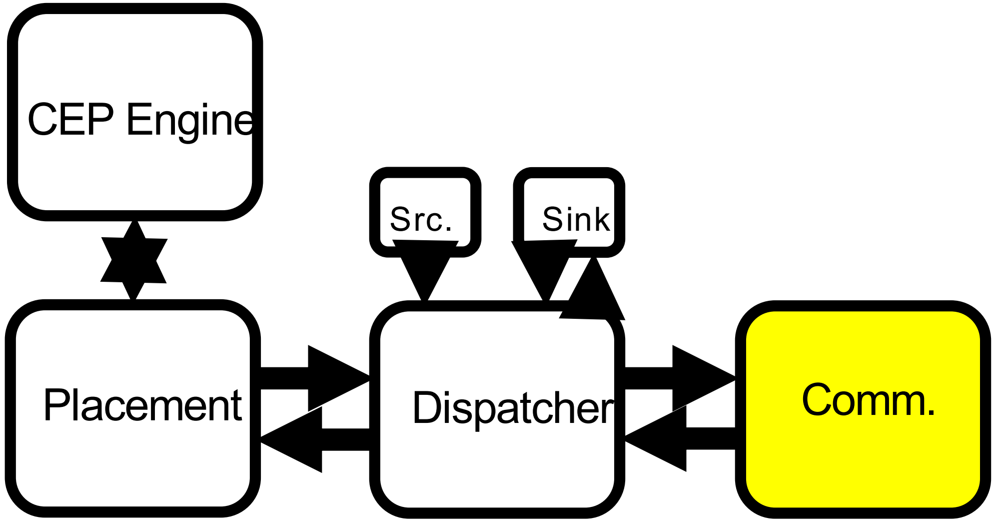
Starks, F., Plagemann, T.: *Operator placement for efficient distributed complex event processing in MANETs*, WiMOB 2015

Placement


Placement Policy
Placement Mechanism
Event routing table

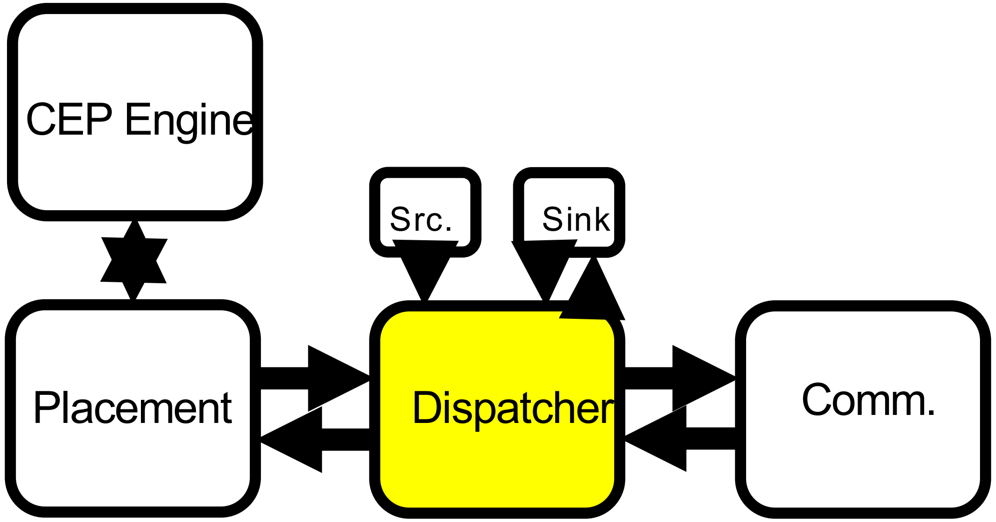
Source and Sink

Communication



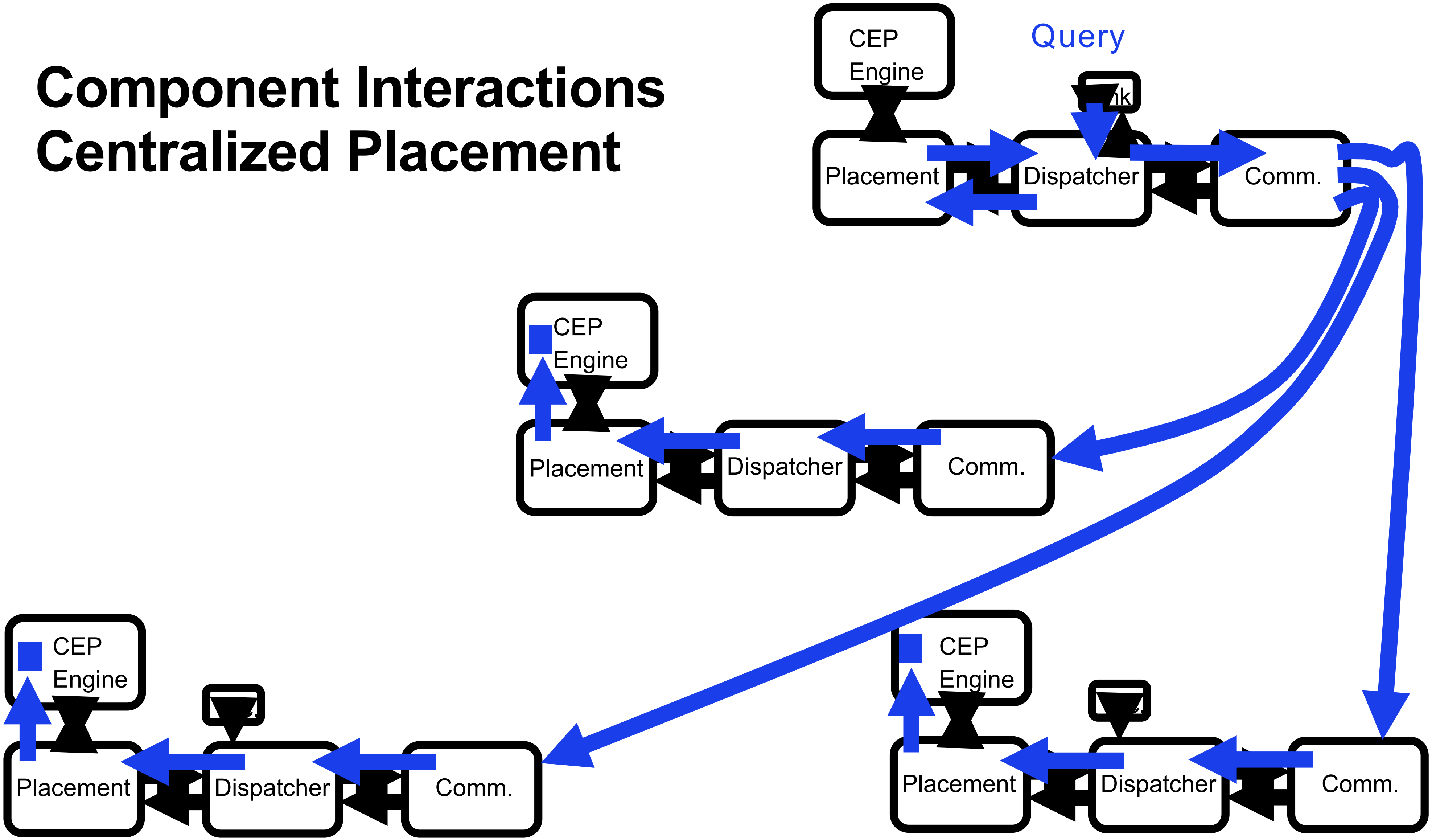
- Responsible for transport of messages
  - Placement messages
    - Forwarding of (parts of) operator graph
    - Coordination of placement adaptation
  - Event notifications
- Current implementation uses UDP

Dispatcher

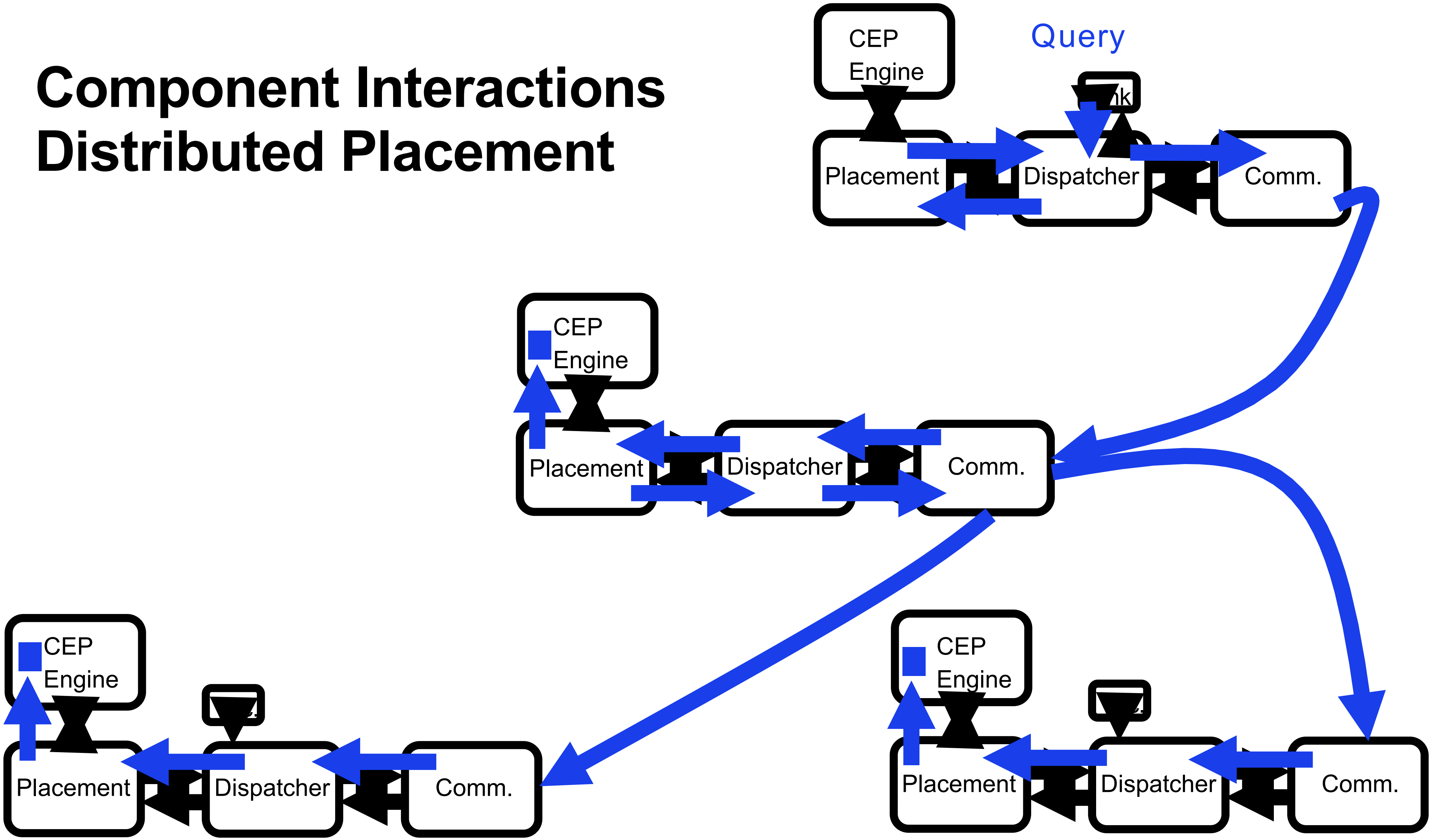


- Facade component
- Dispatches

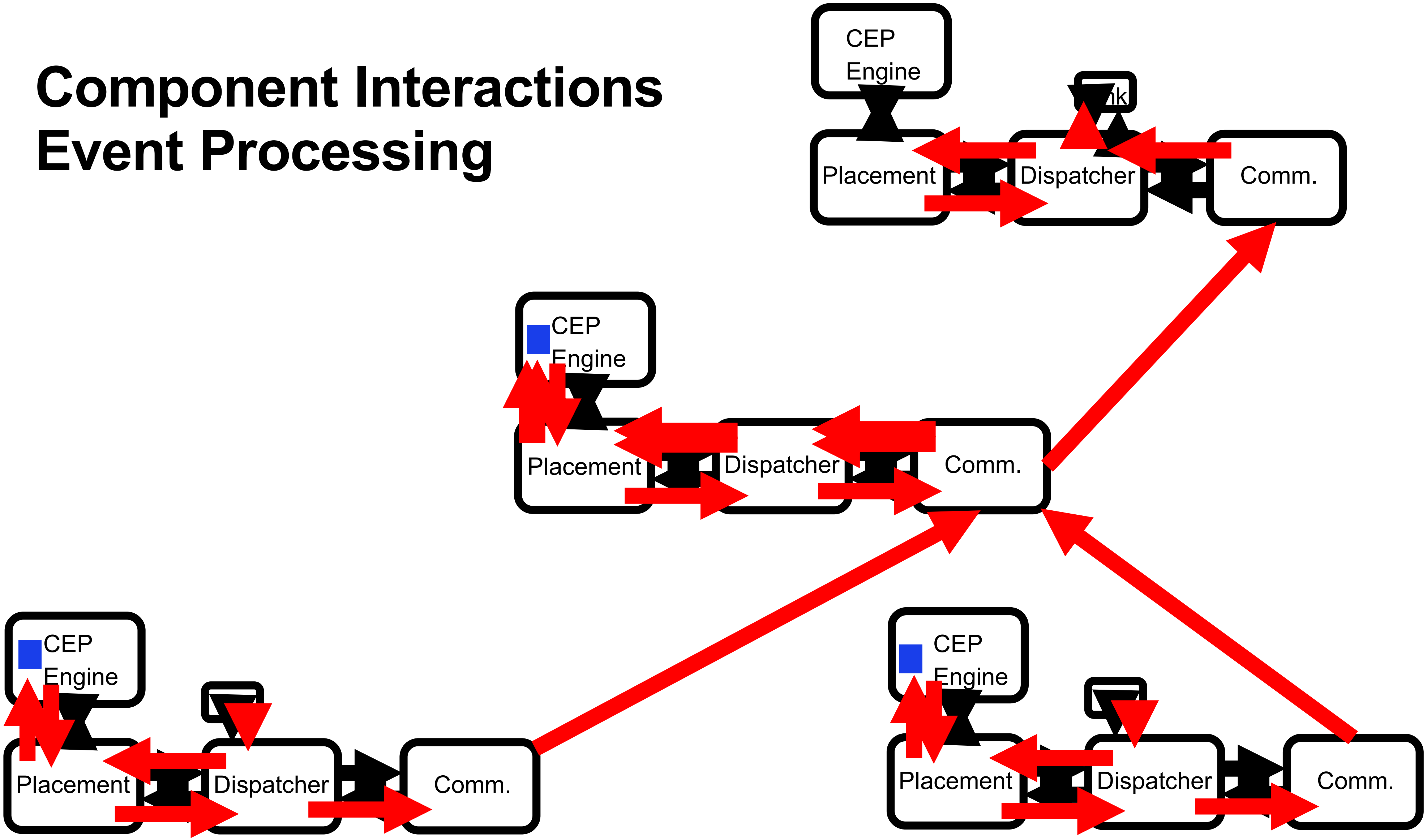
Component Interactions Centralized Placement



Component Interactions Distributed Placement

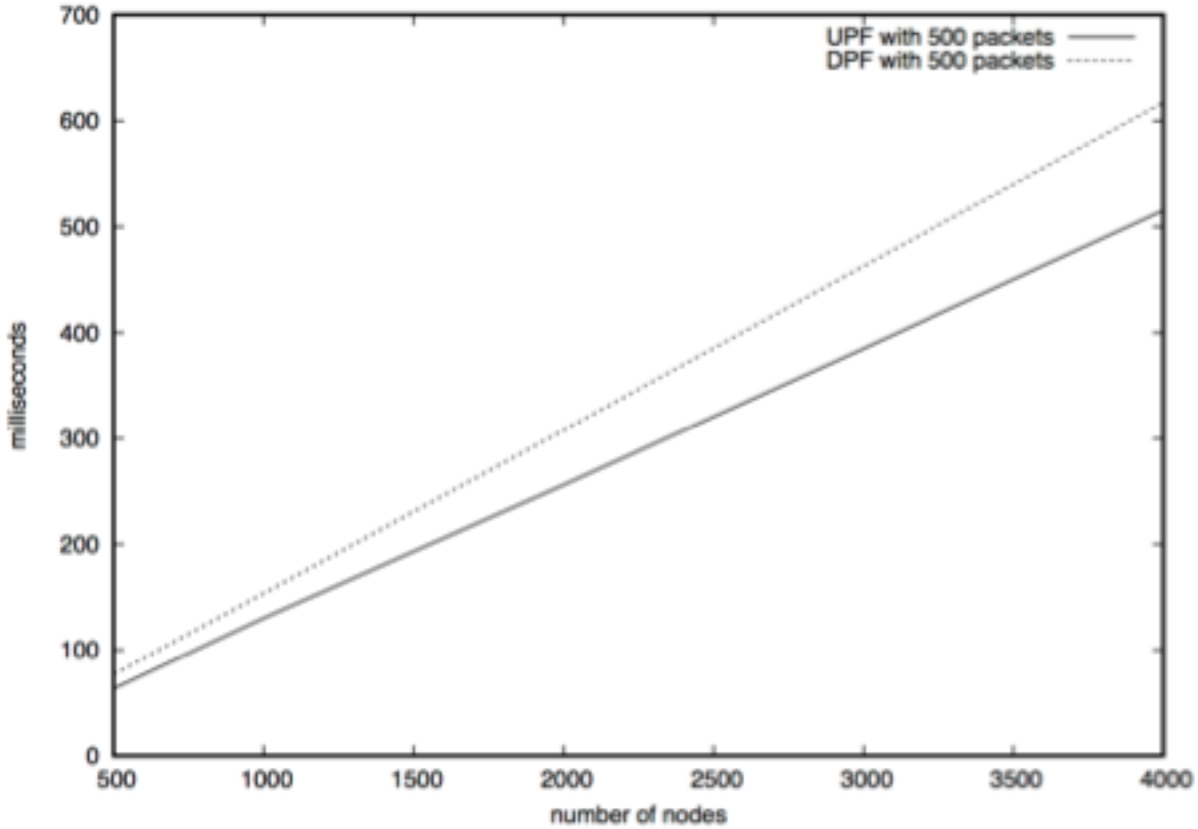


Component Interactions Event Processing



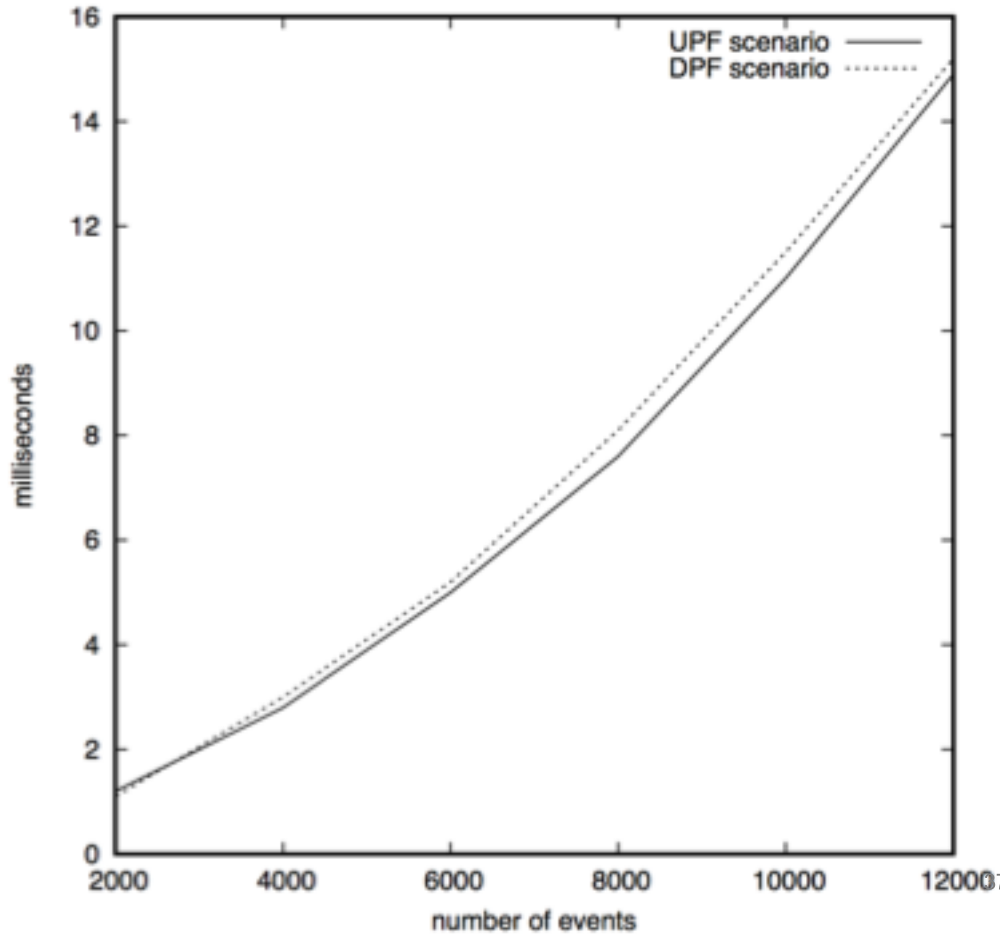
Scalability: number of brokers

Number of brokers		
brokers	events	operators
500	500	1
1000		
2000		
3000		
4000		



Scalability: number of events

Number of events		
events	brokers	operators
2000	1	1
4000		
6000		
8000		
10000		
12000		



End of Part 1

- Very short motivation for distributed CEP
- Design approach
- Components and their responsibility
- Component interaction

- Components correspond to objects in the code (part 3)
- To understand the implementation it is very important to understand ns-3 (next part)

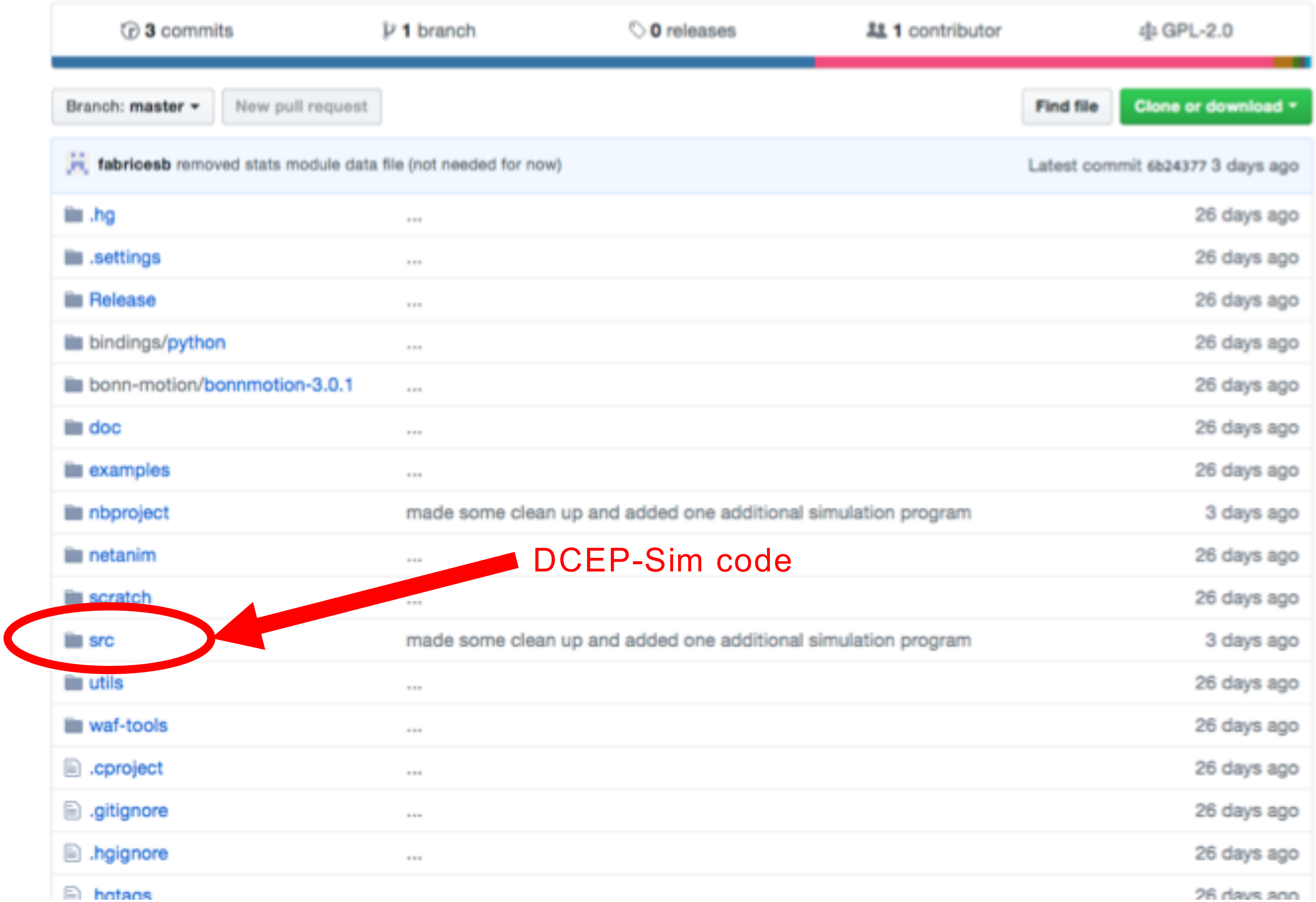
Outline (cont.)

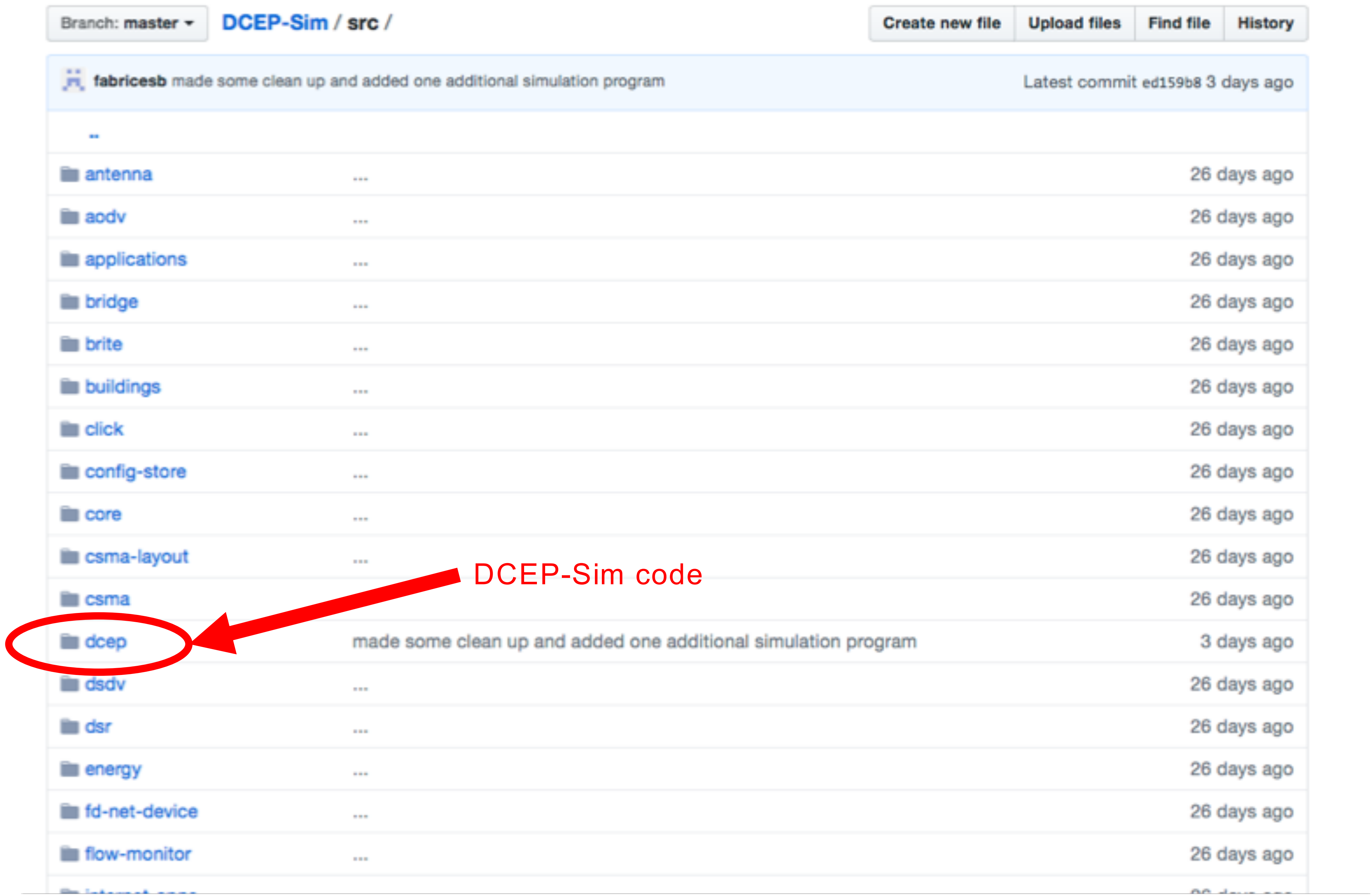
- DCEP-Sim use and extensions
  – Overview code structure
  – How do I run DCEP-Sim & how works a «script»
  – Changing the workload
  – How are placement policies implemented -> adding new placement
  – How are operators implemented -> adding new operators
- Conclusions

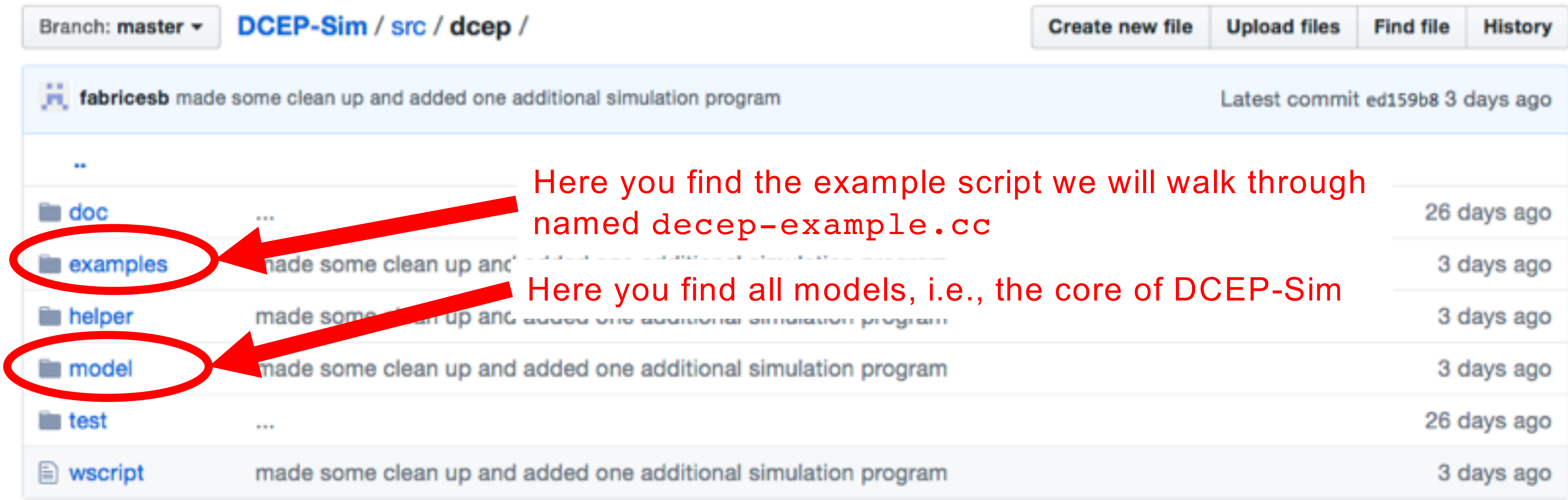
- Hands-on if you want to install ns-3 and run DCEP-Sim on you Linux laptop

DCEP-Sim on github

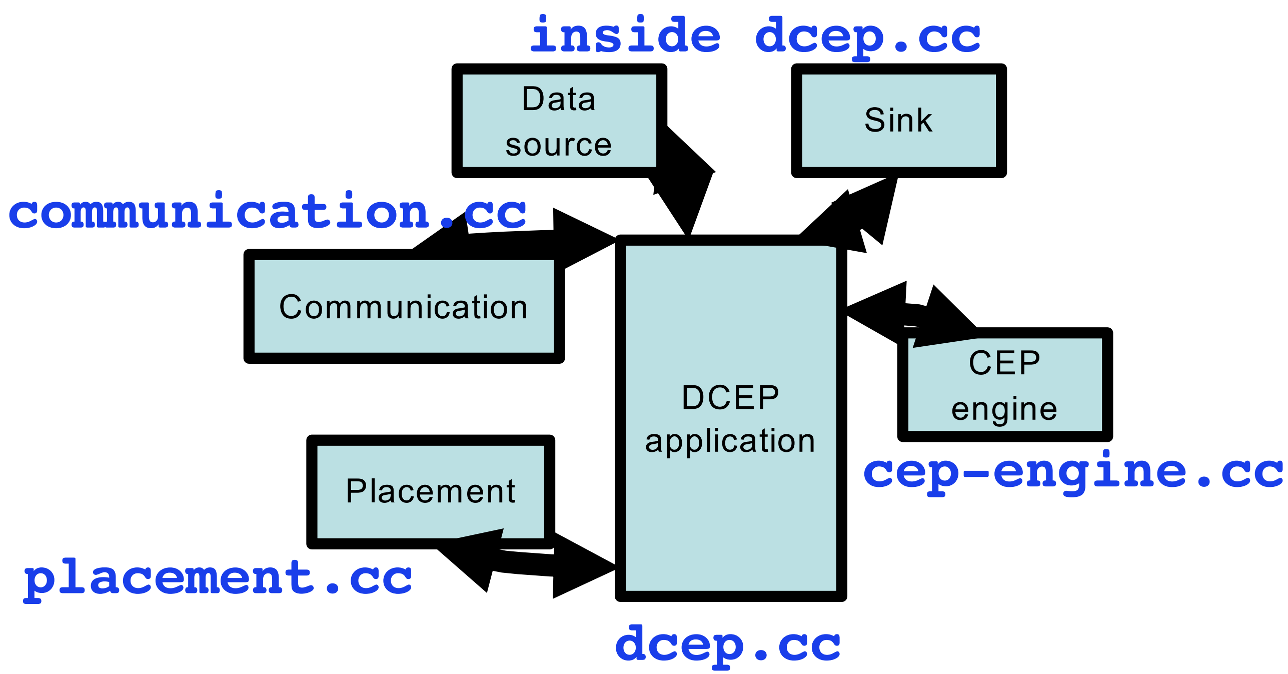
- https://github.com/fabricesb/DCEP-Sim
- GNU GPLv2 license (to be in line with ns-3)


3 commits
1 branch
0 releases
1 contributor
GPL-2.0
Branch: master
New pull request
Find file
Clone or download
fabricesb removed stats module data file (not needed for now)
Latest commit 6b24377 3 days ago
.hg
.settings
Release
bindings/python
bonn-motion/bonnmotion-3.0.1
doc
examples
nbproject
made some clean up and added one additional simulation program
netanim
scratch
src
made some clean up and added one additional simulation program
utils
waf-tools
.cproject
.gitignore
.hgignore
26 days ago
3 days ago
DCEP-Sim code


Branch: master
DCEP-Sim / src /
Create new file
Upload files
Find file
History
fabricesb made some clean up and added one additional simulation program
Latest commit ed159b8 3 days ago
..
antenna ... 26 days ago
aodv ... 26 days ago
applications ... 26 days ago
bridge ... 26 days ago
brite ... 26 days ago
buildings ... 26 days ago
click ... 26 days ago
config-store ... 26 days ago
core ... 26 days ago
csma-layout ... 26 days ago
csma ... 26 days ago
dcep made some clean up and added one additional simulation program 3 days ago
dsdv ... 26 days ago
dsr ... 26 days ago
energy ... 26 days ago
fd-net-device ... 26 days ago
flow-monitor ... 26 days ago
DCEP-Sim code

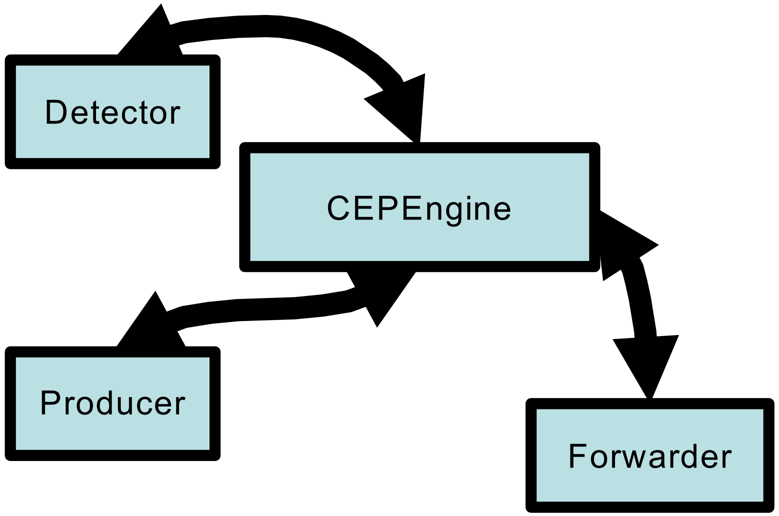

Branch: master
DCEP-Sim / src / dcep /
Create new file
Upload files
Find file
History
fabricesb made some clean up and added one additional simulation program
Latest commit ed159b8 3 days ago
..
doc
...
26 days ago
examples
made some clean up and added one additional simulation program
3 days ago
helper
made some clean up and added one additional simulation program
3 days ago
model
made some clean up and added one additional simulation program
3 days ago
test
...
26 days ago
wscript
made some clean up and added one additional simulation program
3 days ago
Here you find the example script we will walk through named `decep-example.cc`
Here you find all models, i.e., the core of DCEP-Sim

Components, objects, and aggregation



Components, objects, and aggregation (cont.)

All in cep-engine.cc



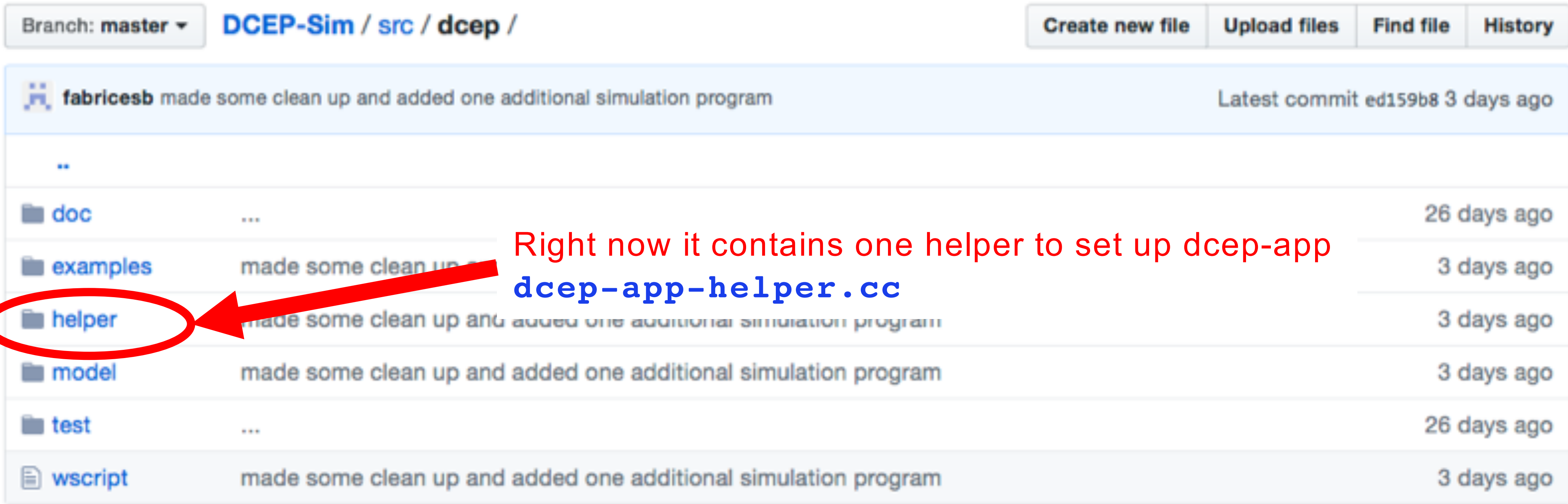

```

CEPEngine::CEPEngine()
{
    Ptr<Forwarder> forwarder = CreateObject<Forwarder>();
    Ptr<Detector> detector = CreateObject<Detector>();
    Ptr<Producer> producer = CreateObject<Producer>();
    AggregateObject(forwarder);
    AggregateObject(detector);
    AggregateObject(producer);


}

```

Branch: master ▾ **DCEP-Sim / src / dcep / model /**

Create new file | Upload files | Find file | History

fabricesb made some clean up and added one additional simulation program — Latest commit ed159b8 3 days ago

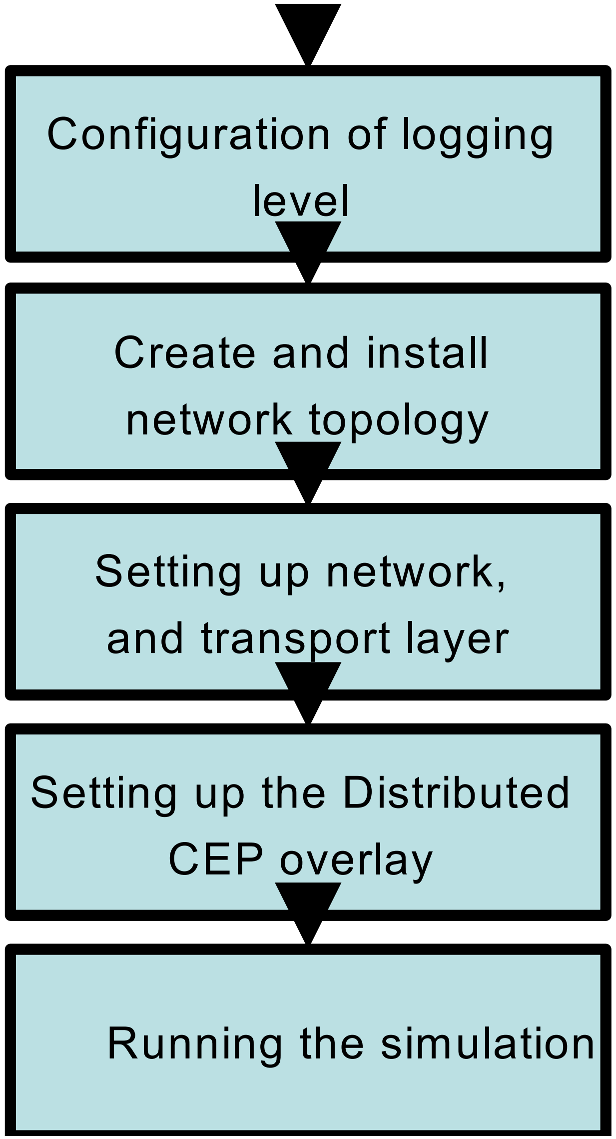
..		
cep-engine.cc	made some clean up and added one additional simulation program	3 days ago
cep-engine.h	...	26 days ago
common.h	...	26 days ago
communication.cc	made some clean up and added one additional simulation program	3 days ago
communication.h	made some clean up and added one additional simulation program	3 days ago
dcep-header.cc	...	26 days ago
dcep-header.h	...	26 days ago
dcep-state.cc	...	26 days ago
dcep-state.h	...	26 days ago
dcep.cc	made some clean up and added one additional simulation program	3 days ago
dcep.h	made some clean up and added one additional simulation program	3 days ago
message-types.h	...	26 days ago
placement.cc	made some clean up and added one additional simulation program	3 days ago
placement.h	...	26 days ago
resource-manager.cc	...	26 days ago
resource-manager.h	...	26 days ago
seq-ts-header.cc	...	26 days ago
seq-ts-header.h	...	26 days ago

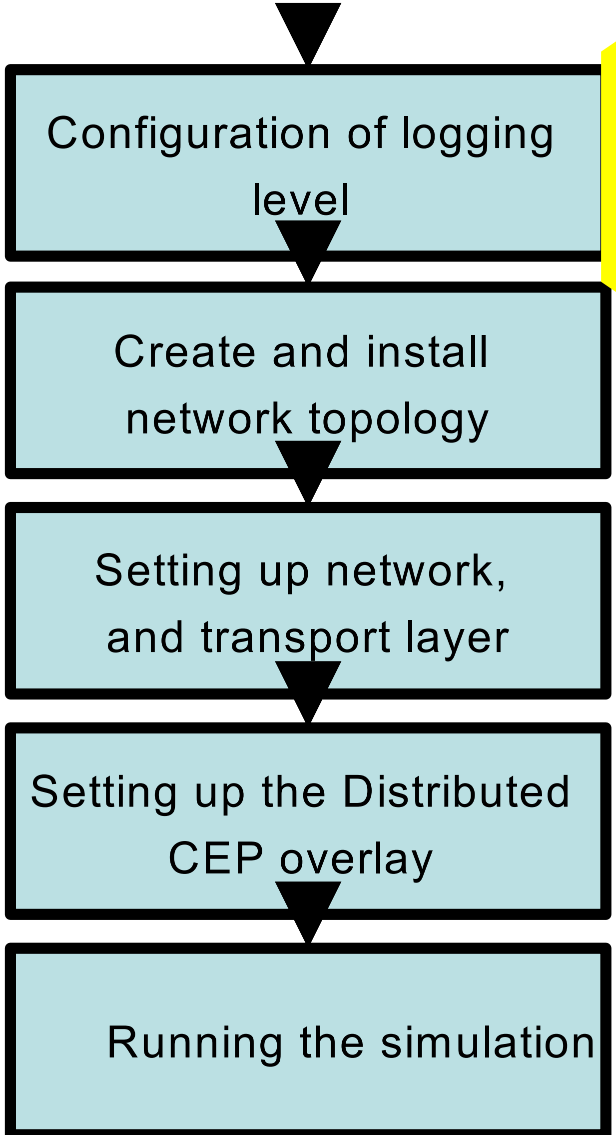


Right now it contains one helper to set up dcep-app

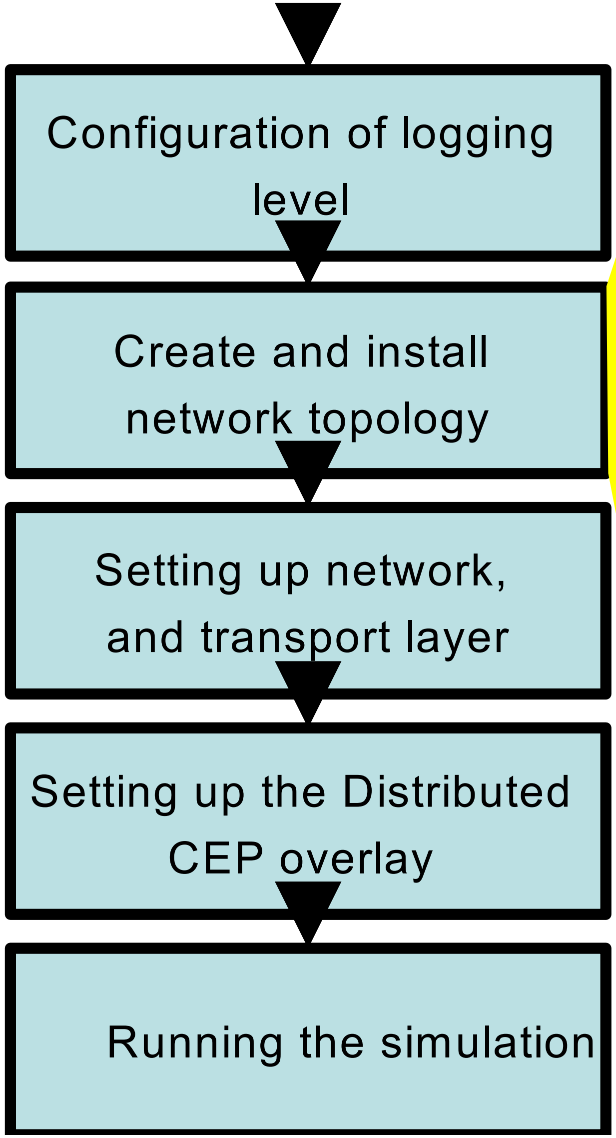
`dcep-app-helper.cc`

Typical elements of a script






```
LogComponentEnable ("Placement", LOG_LEVEL_INFO);
LogComponentEnable ("Dcep", LOG_LEVEL_INFO);
LogComponentEnable ("Detector", LOG_LEVEL_INFO);
LogComponentEnable ("Communication",
                                        LOG_LEVEL_INFO);
```

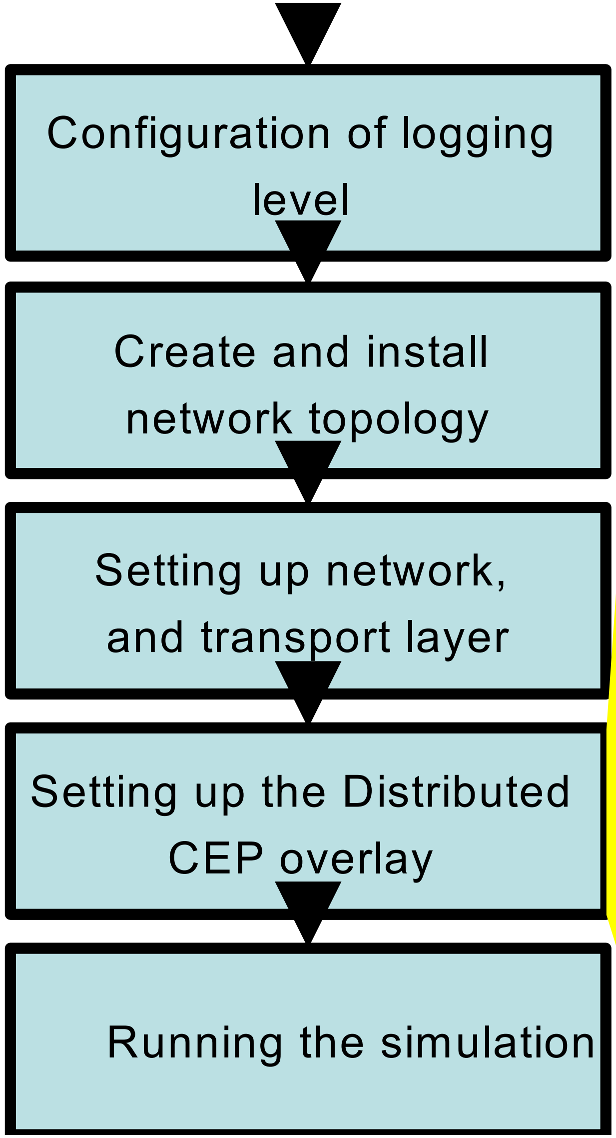



```
uint32_t numNodes = gridWidth*gridWidth;
NodeContainer n;
n.Create (numNodes);

NetDeviceContainer devices =
                         SetupWirelessNetwork(n);
MobilityHelper mobility;

mobility.SetPositionAllocator
  ("ns3::GridPositionAllocator", "MinX",
    DoubleValue (0.0), "MinY", DoubleValue (0.0),
   "DeltaX", DoubleValue (distance), "DeltaY",
    DoubleValue (distance), "GridWidth",
    UintegerValue (gridWidth), "LayoutType",
    StringValue ("RowFirst"));

mobility.SetMobilityModel
  ("ns3::ConstantPositionMobilityModel");
mobility.Install (n);
```

Typical elements of a script

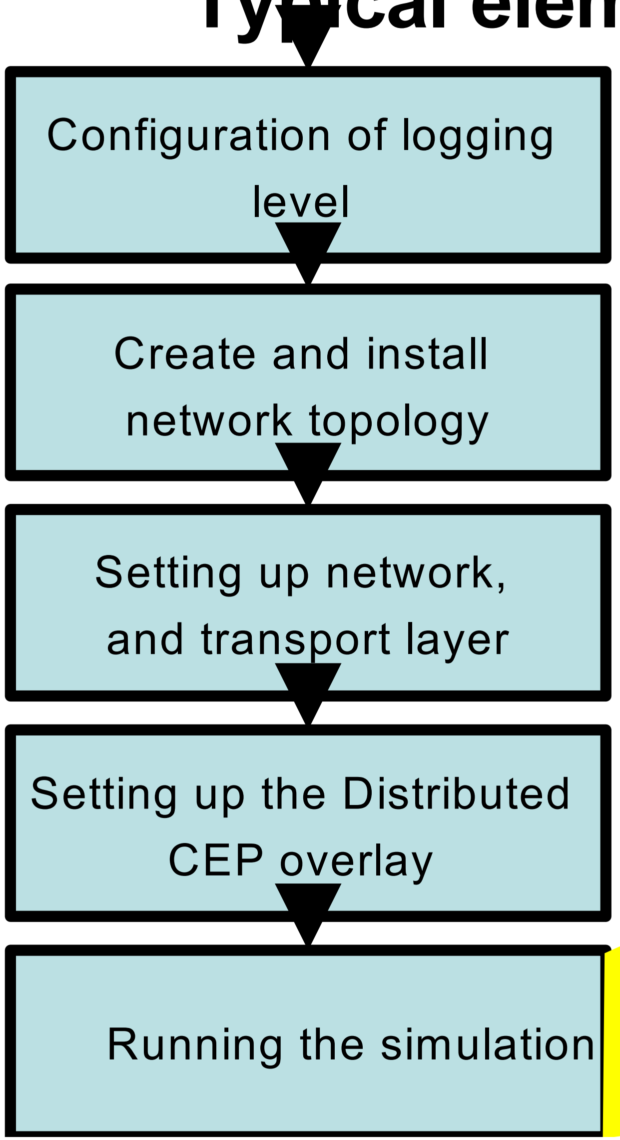
- Configuration of logging level
- Create and install network topology
- Setting up network, and transport layer
- Setting up the Distributed CEP overlay
- Running the simulation

```
OlsrHelper olsr;
InternetStackHelper internet;
internet.SetRoutingHelper (olsr);
internet.Install (n);
Ipv4AddressHelper ipv4;
ipv4.SetBase ("10.1.1.0", "255.255.255.0");
Ipv4InterfaceContainer iface =
    ipv4.Assign (devices);
```




```
sinkAddress = Address(iface.GetAddress (0));
DcepAppHelper dcepApphelper;
ApplicationContainer dcepApps =
   dcepApphelper.Install (n);
uint32_t eventCode = 1;

for(uint32_t i = 0; i <= numNodes; i++) {
     dcepApps.Get(i)->SetAttribute
  ("SinkAddress", AddressValue(sinkAddress));
     dcepApps.Get(i)->SetAttribute("placement
     policy", StringValue(placementPolicy));

     if(i == 0) {  /* sink node*/
           dcepApps.Get(i)->SetAttribute
           ("IsSink", BooleanValue(true));
     }
     else if ((i == (numNodes-1)) || (i == (numNodes-2))){
          dcepApps.Get(i)->SetAttribute("IsGenerator",
                BooleanValue(true));
          dcepApps.Get(i)->SetAttribute("event code",
                UintegerValue (eventCode++));
          dcepApps.Get(i)->SetAttribute("number of
                events", UintegerValue (numberOfEvents));
    }
}
```

Typical elements of a script




```
dcepApps.Start (Seconds (1.0));
dcepApps.Stop (Seconds (30.0));
Simulator::Stop(Seconds(35.0));
Simulator::Run ();
Simulator::Destroy ();
```

Change the workload

- Current event sources produce uniform traffic
- Configure Distributed CEP instances as data sources in the script, e.g.,

```
dcepApps.Get(0)->SetAttribute("IsGenerator", BooleanValue(true));
dcepApps.Get(0)->SetAttribute("event code",UintegerValue (eventCode++));
```

- Set number of events in the script

```
dcepApps.Get(i)->SetAttribute("number of events",
                                                    UintegerValue (numberOfEvents));
```

Change the workload (cont.)

- Currently, the event rate is set in the `DataSource::GenerateAtomicEvents()` implementation in `dcep.cc`

```
if(counter < numEvents)
{
          Simulator::Schedule (MilliSeconds (100),
                              &DataSource::GenerateAtomicEvents, this);
}
```

Good example of scheduling discrete ns-3 events….

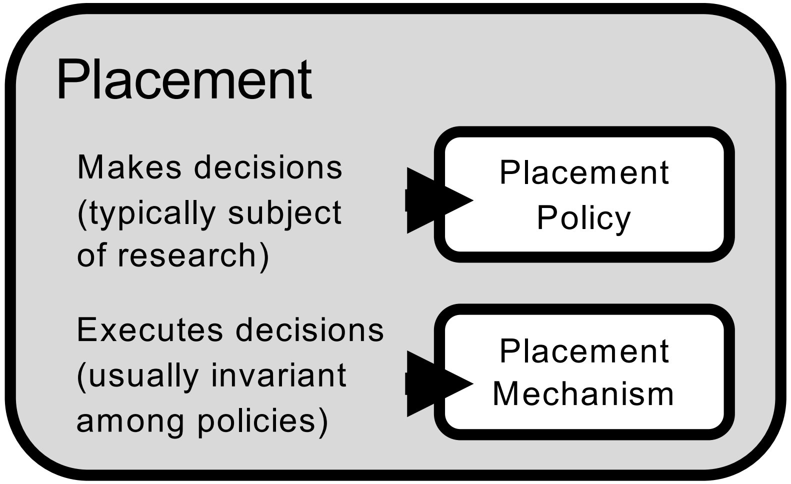
….. to generate at a fixed rate atomic events!

Change workload (cont.)

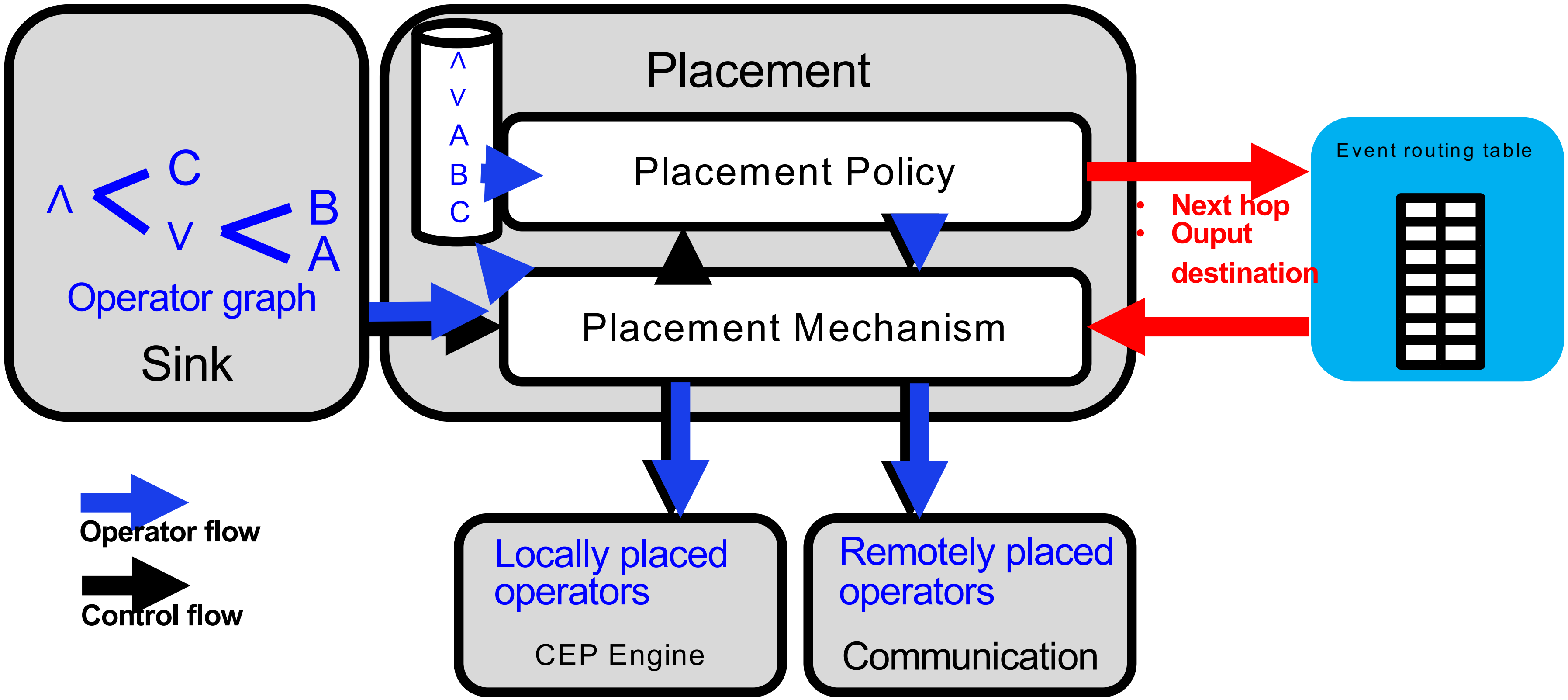
- For more complex event patterns extend the data source model or create a new data source model
  - Get inspired by ns-3 traffic models
    - Statistical distributions
    - Trace files
    - ....
  - Extend/modify the function `GenerateCEPEvents()` which can be found in the file `dcep.cc`

Creating Your Own Placement Policy

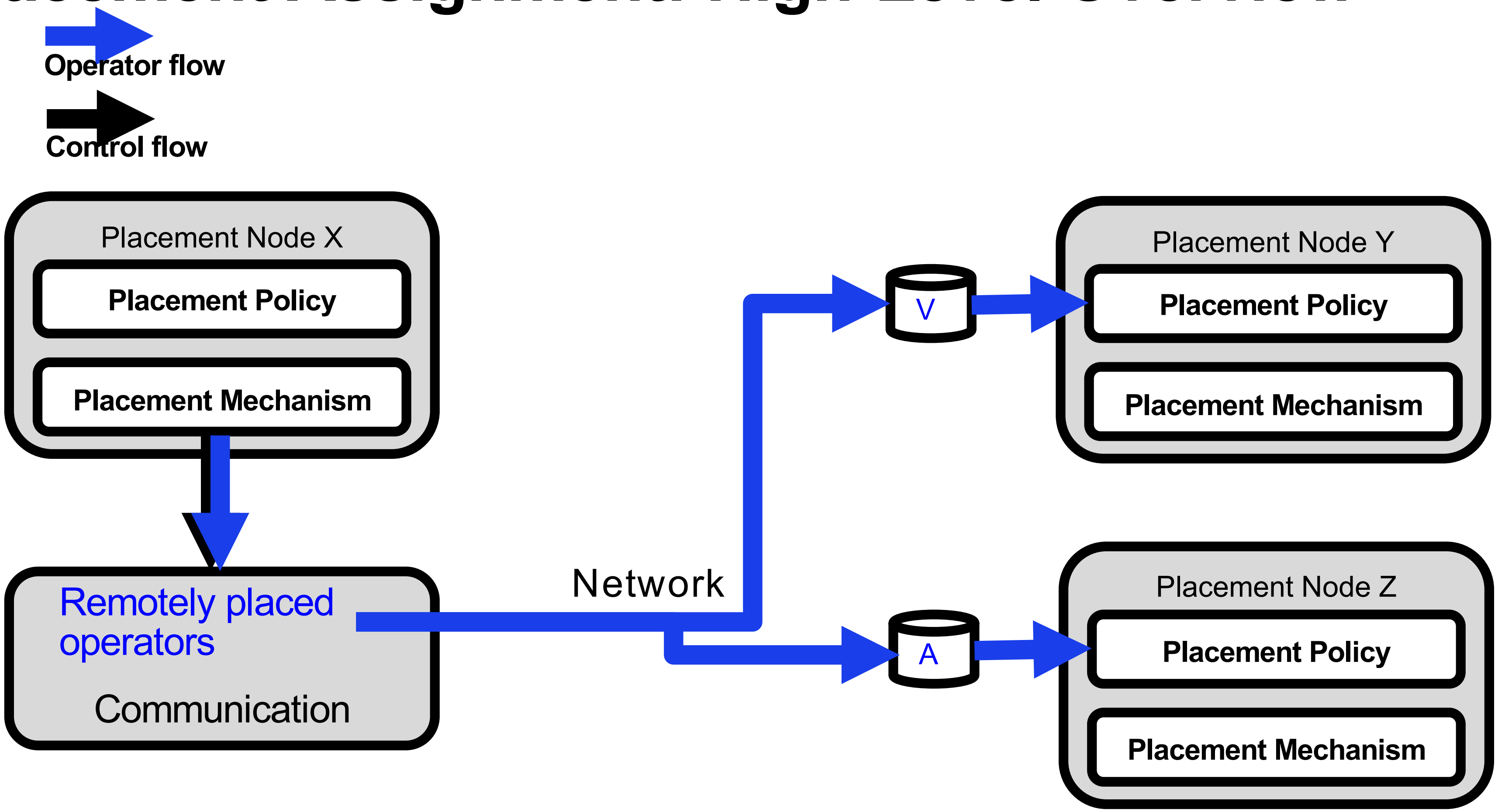
- Main responsabilities of placement:
  1. Operator assignment
  2. Event routing and forwarding
- Approach:
  - High + low level views
  - Creating a new placement policy
  - Example: centralized placement



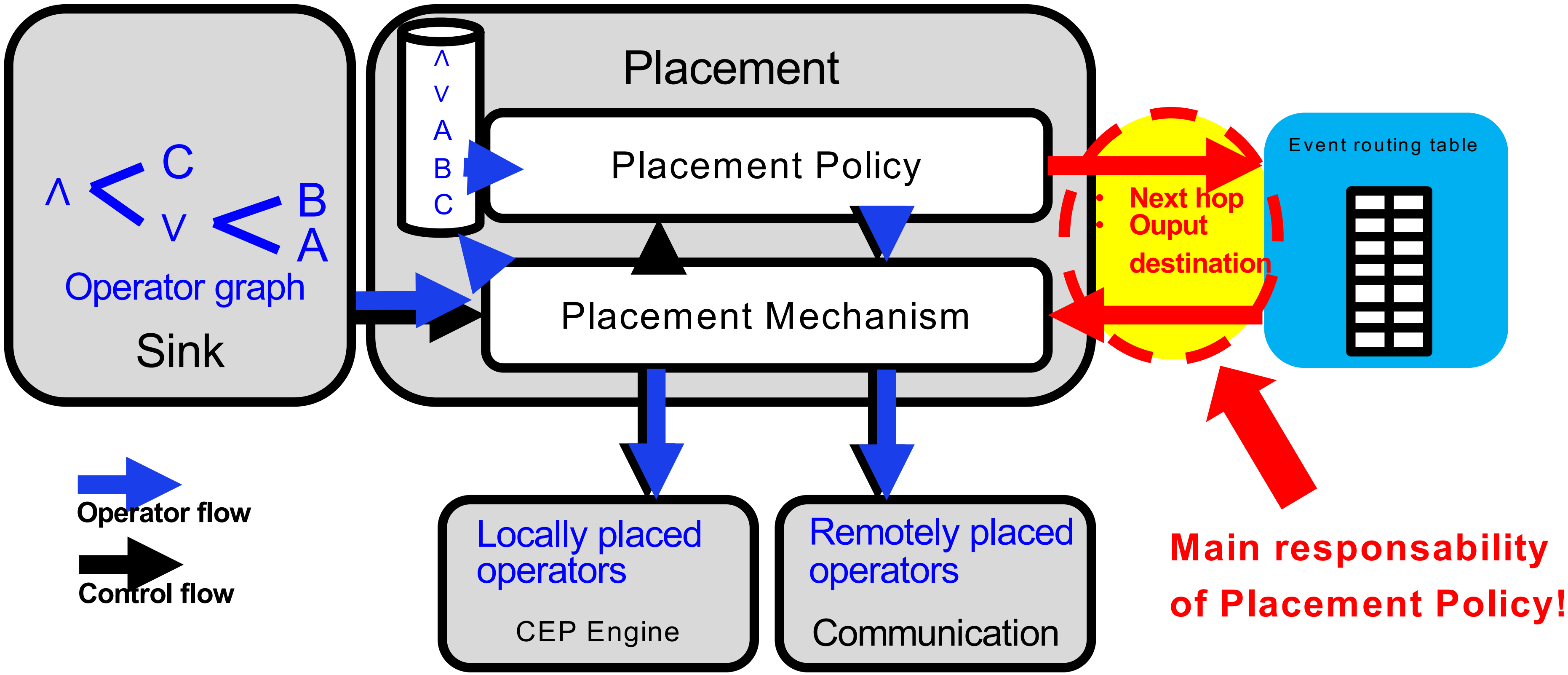
Placement Assignment: High-Level Overview



Placement Assignment: High-Level Overview

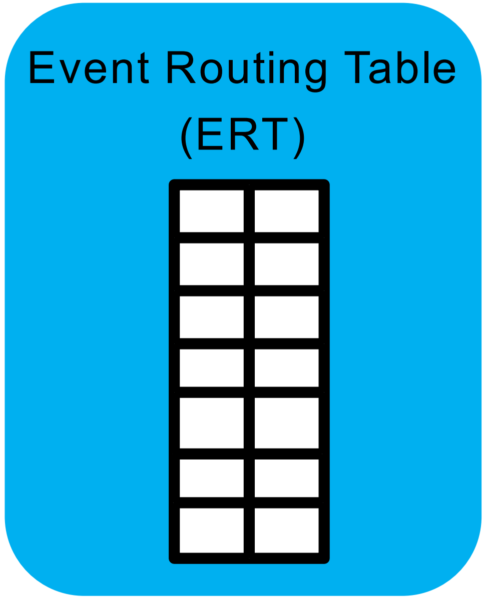


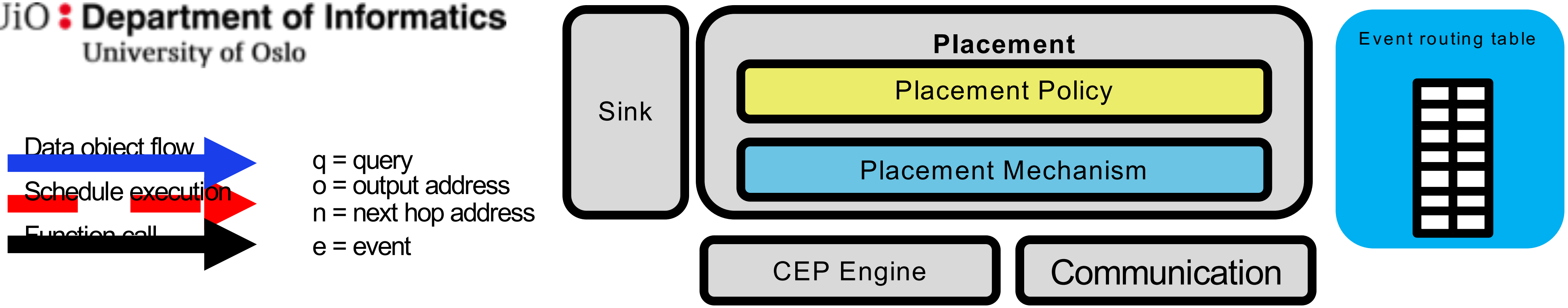
Placement Assignment: High-Level Overview

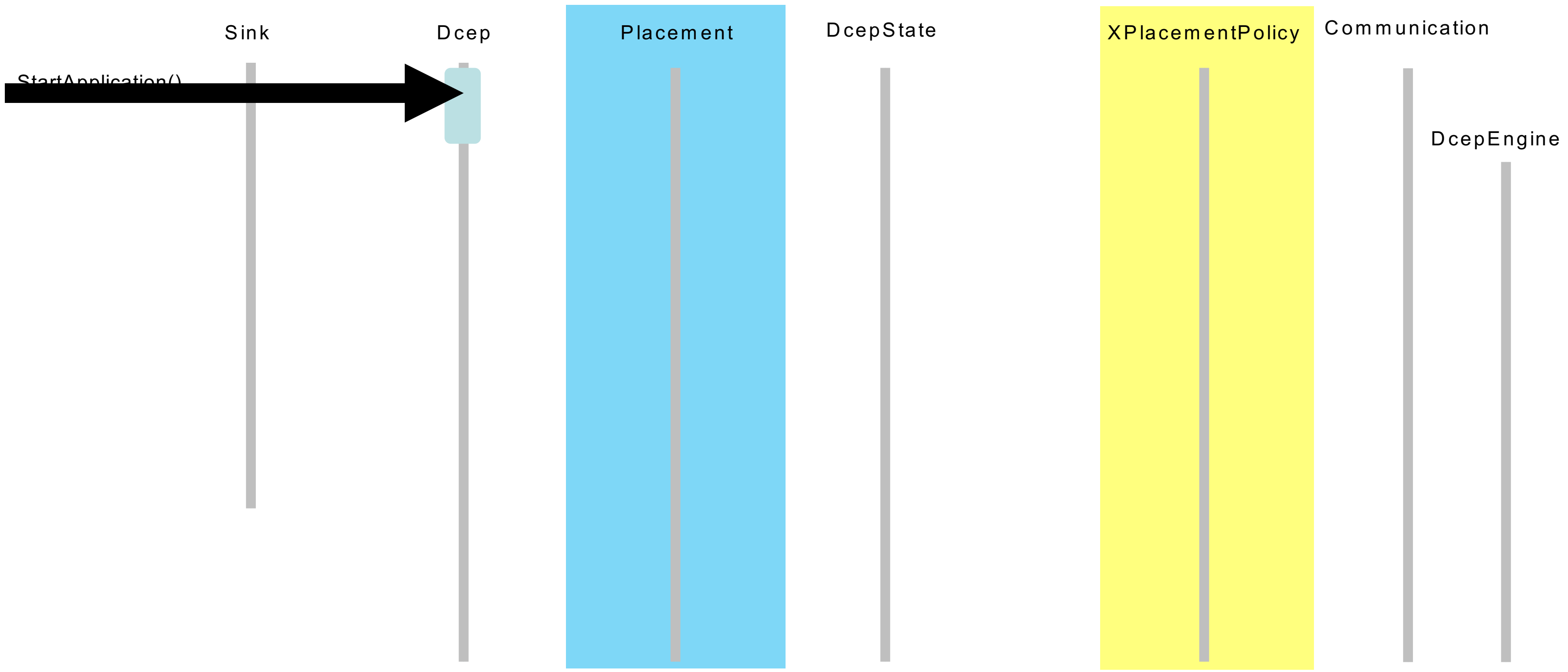


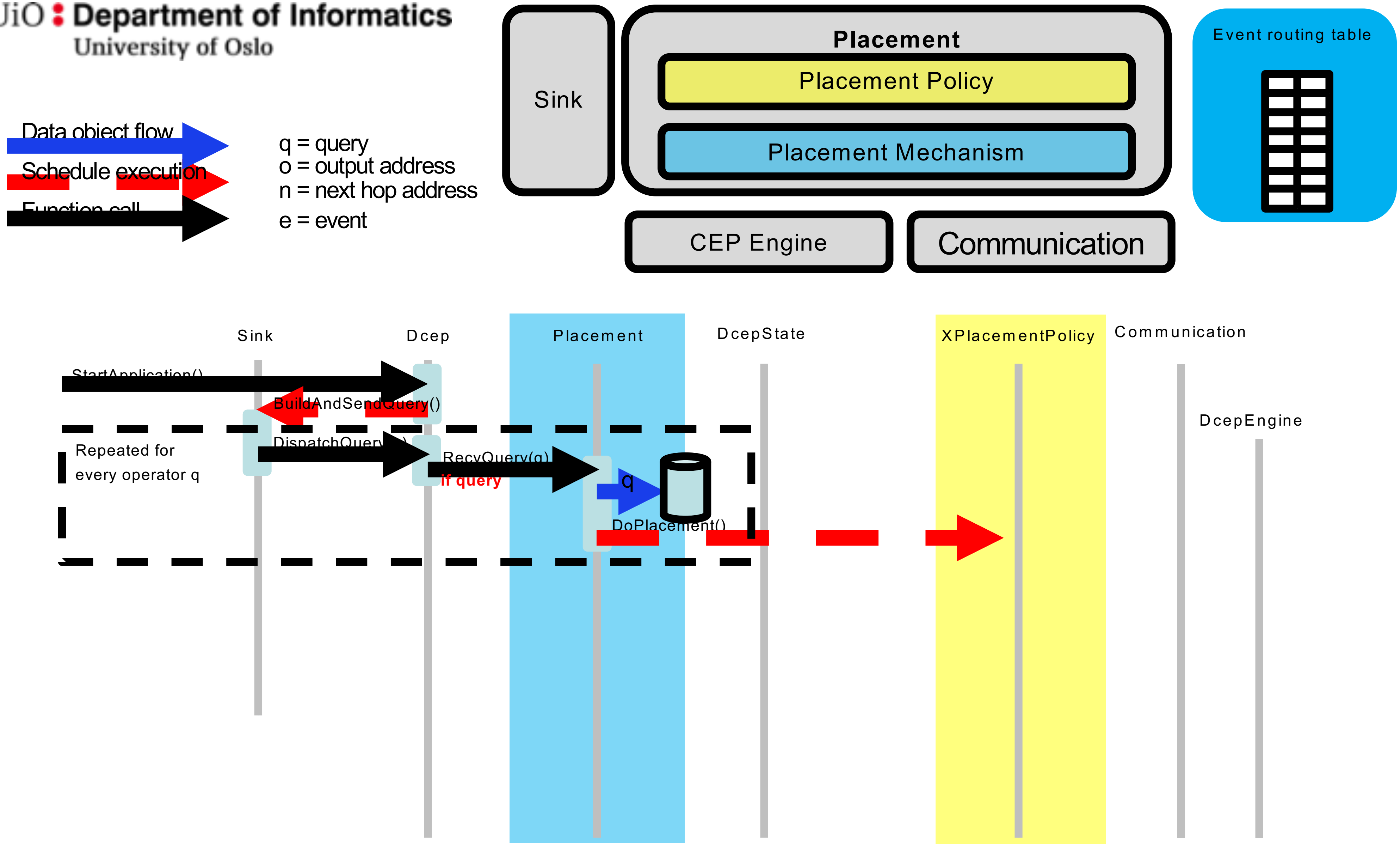
The Event Routing Table (ERT)

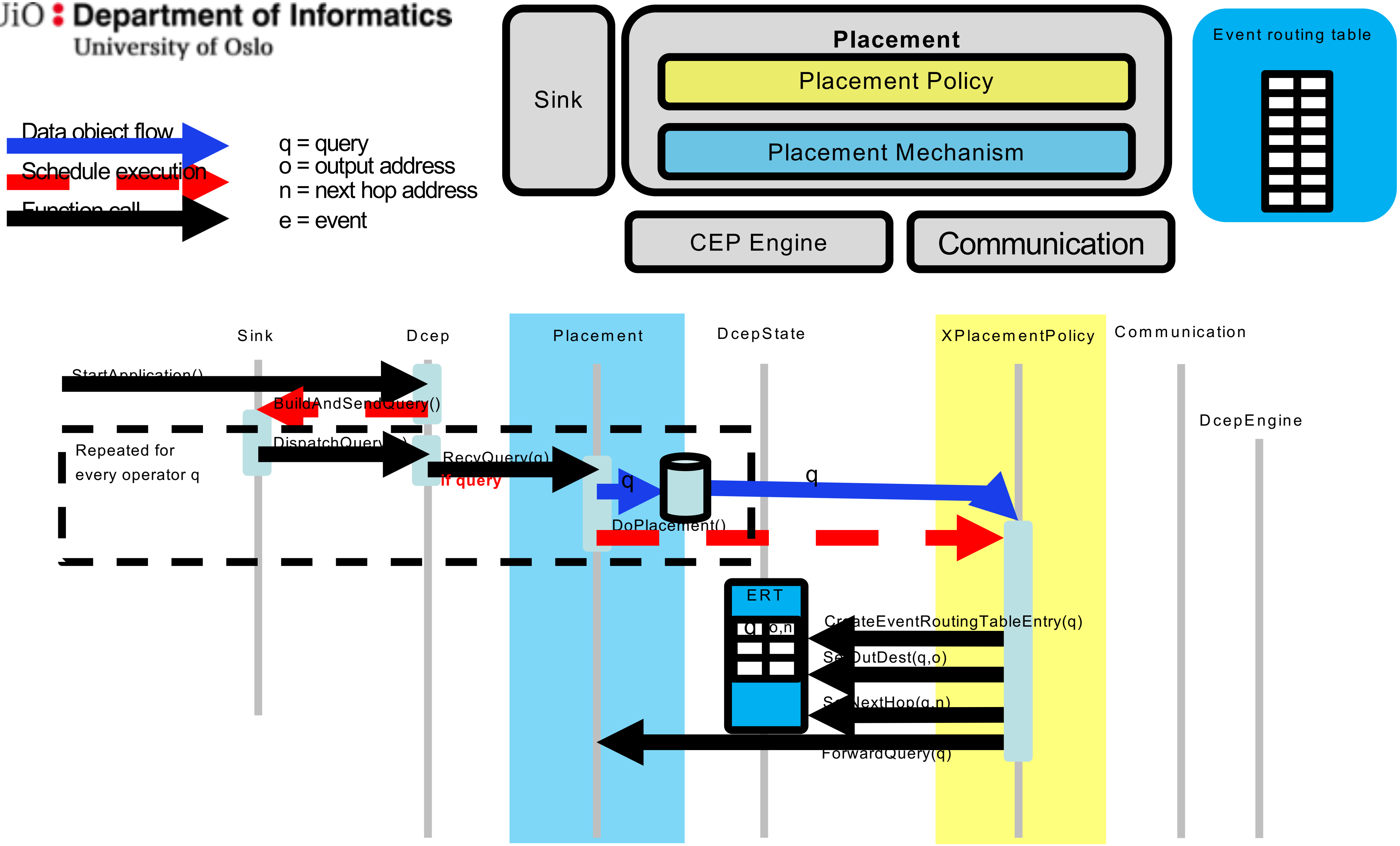
- Accessed via interface called DcepState
- Important fields in entries:
  - **Destination of event (output destination)**
  - **Destination of the query (next hop)**
  - Data sources
- Additional fields mostly for adaptation and monitoring
  - Operator state (active or not)
  - Freeze acknowledgement counter
  - Freeze queue
  - Monitoring
  - Current processor

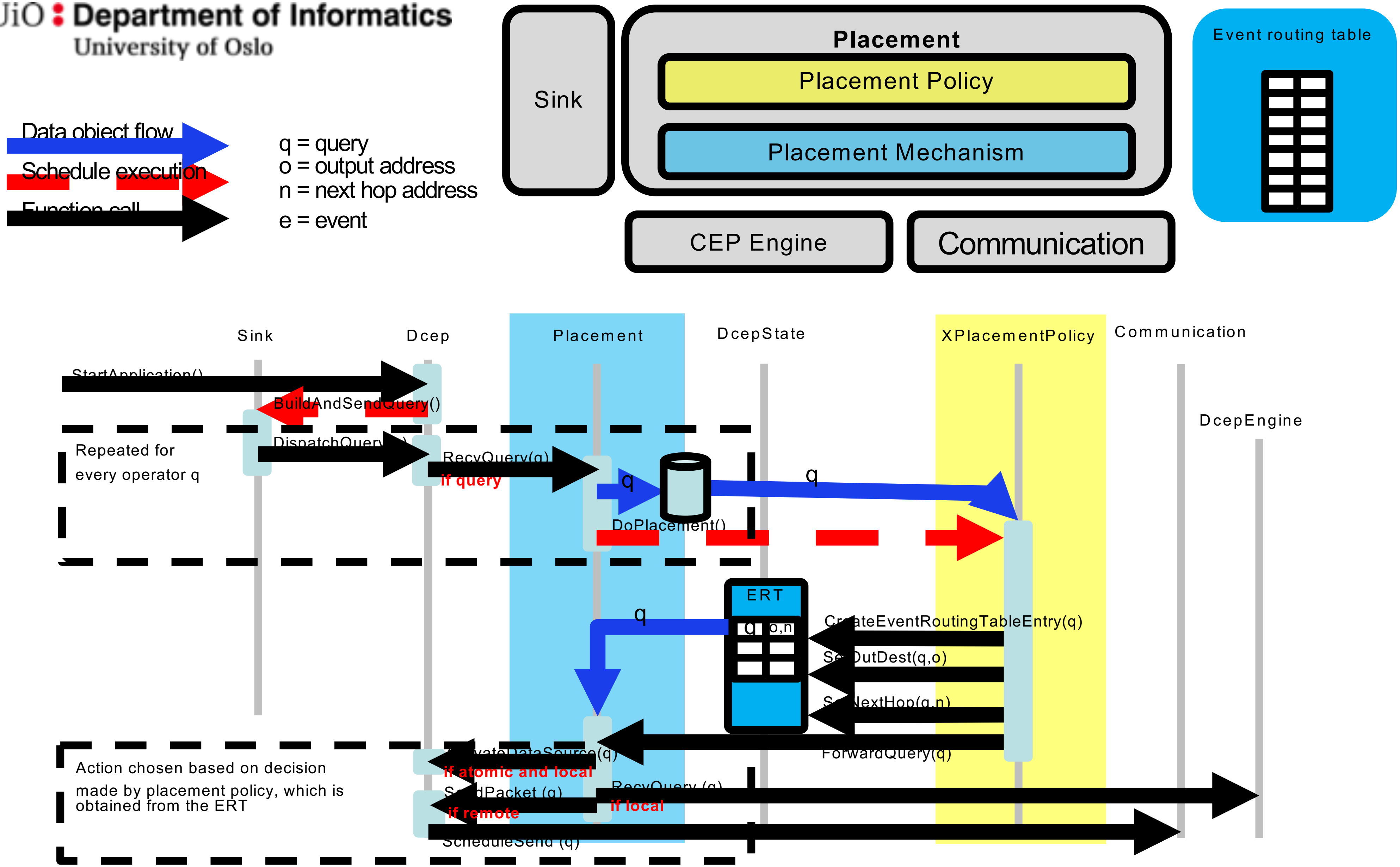


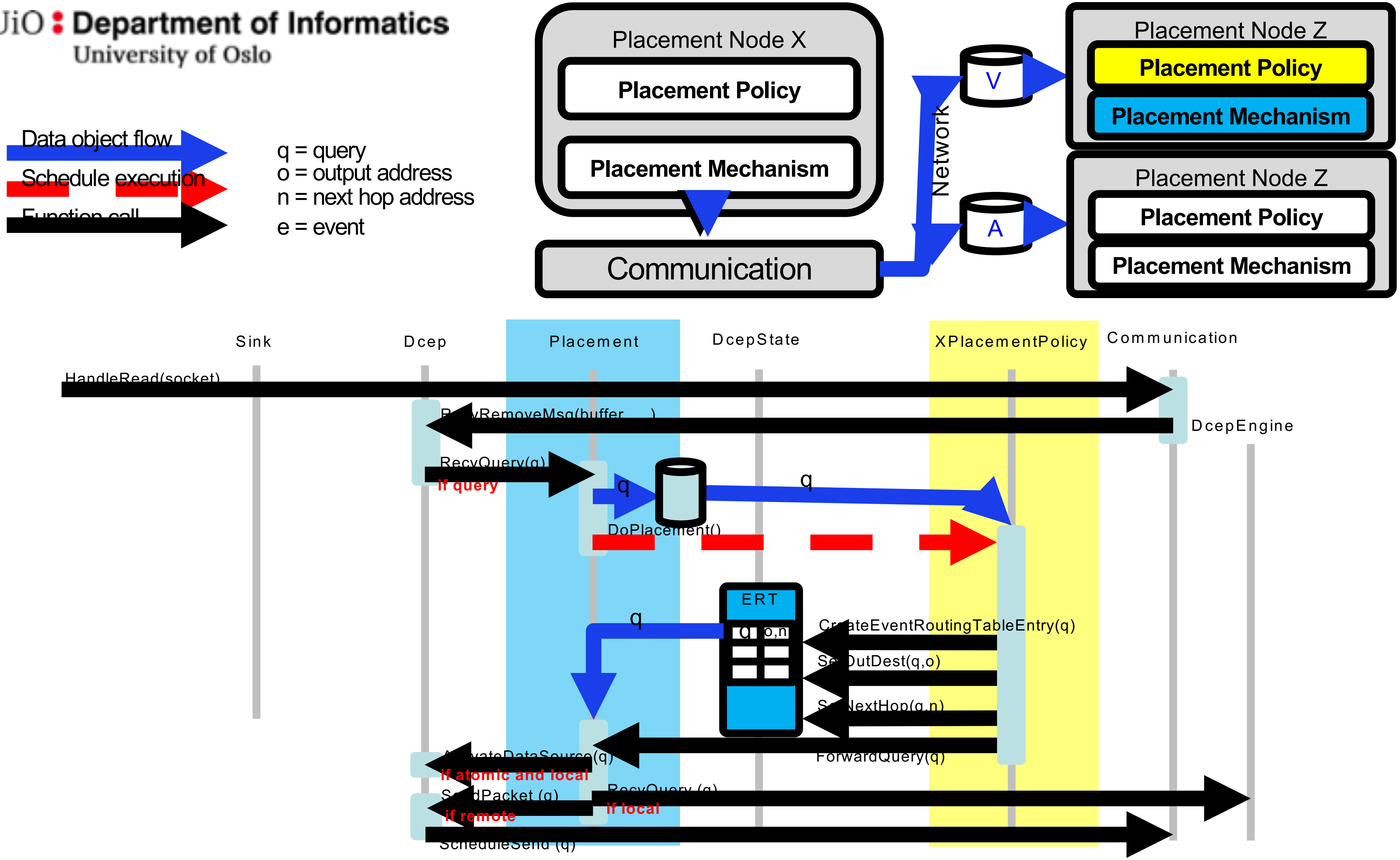

Data object flow
Schedule execution
Function call
q = query
o = output address
n = next hop address
e = event
Sink
Placement
Placement Policy
Placement Mechanism
Event routing table
CEP Engine
Communication


Sink
Dcep
Placement
DcepState
XPlacementPolicy
Communication
DcepEngine
StartApplication()

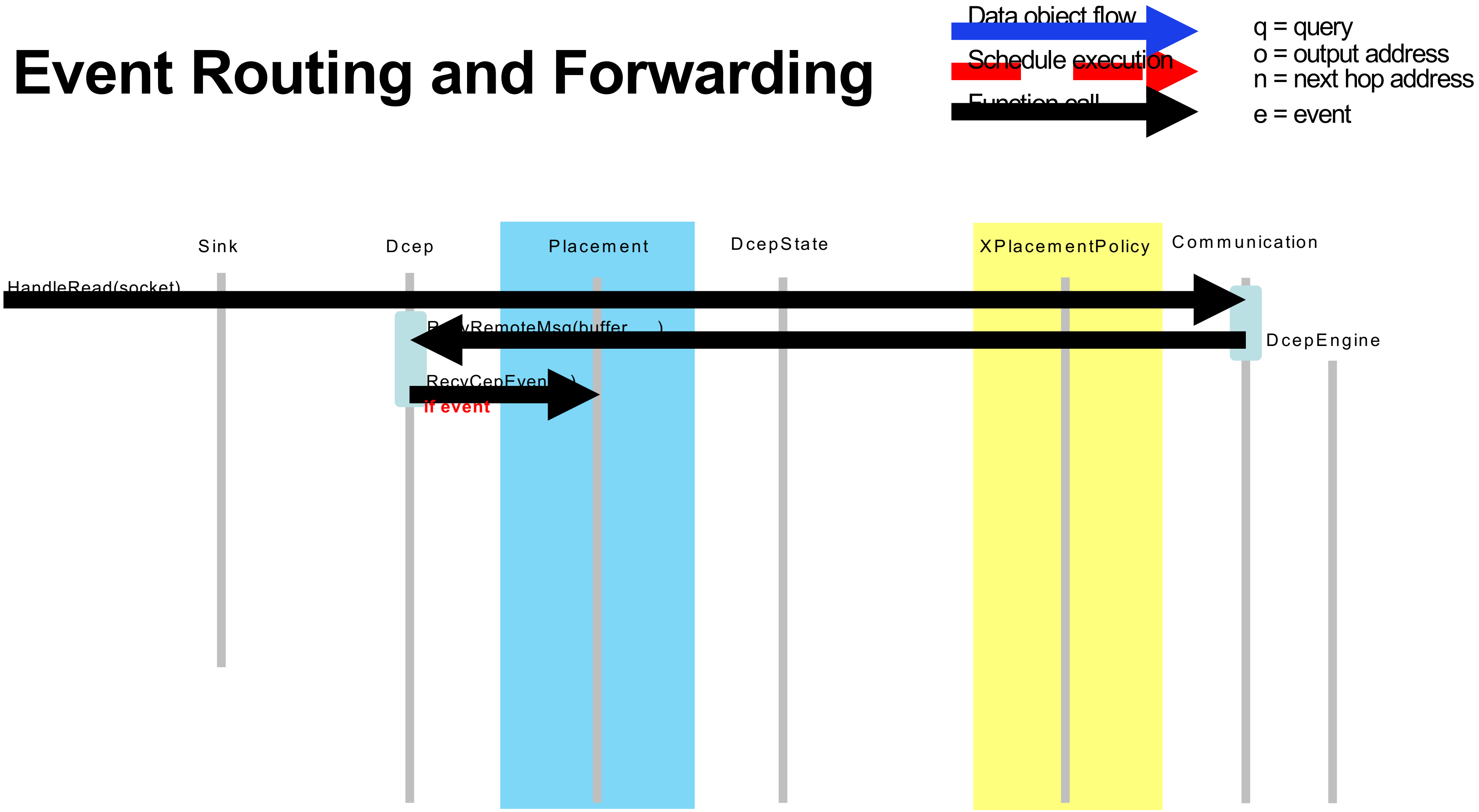

Data object flow
Schedule execution
Function call
q = query
o = output address
n = next hop address
e = event
Sink
Placement
Placement Policy
Placement Mechanism
Event routing table
CEP Engine
Communication
Sink
Dcep
Placement
DcepState
XPlacementPolicy
Communication
DcepEngine
StartApplication()
BuildAndSendQuery()
Repeated for
every operator q
DispatchQuery()
RecvQuery(q)
if query
q
DoPlacement()


Data object flow
Schedule execution
Function call
q = query
o = output address
n = next hop address
e = event
Sink
Placement
Placement Policy
Placement Mechanism
Event routing table
CEP Engine
Communication
Sink
Dcep
Placement
DcepState
XPlacementPolicy
Communication
DcepEngine
StartApplication()
BuildAndSendQuery()
Repeated for
every operator q
DispatchQuery()
RecvQuery(q)
if query
q
q
DoPlacement()
ERT
q
o,n
CreateEventRoutingTableEntry(q)
SetOutDest(q,o)
SetNextHop(q,n)
ForwardQuery(q)

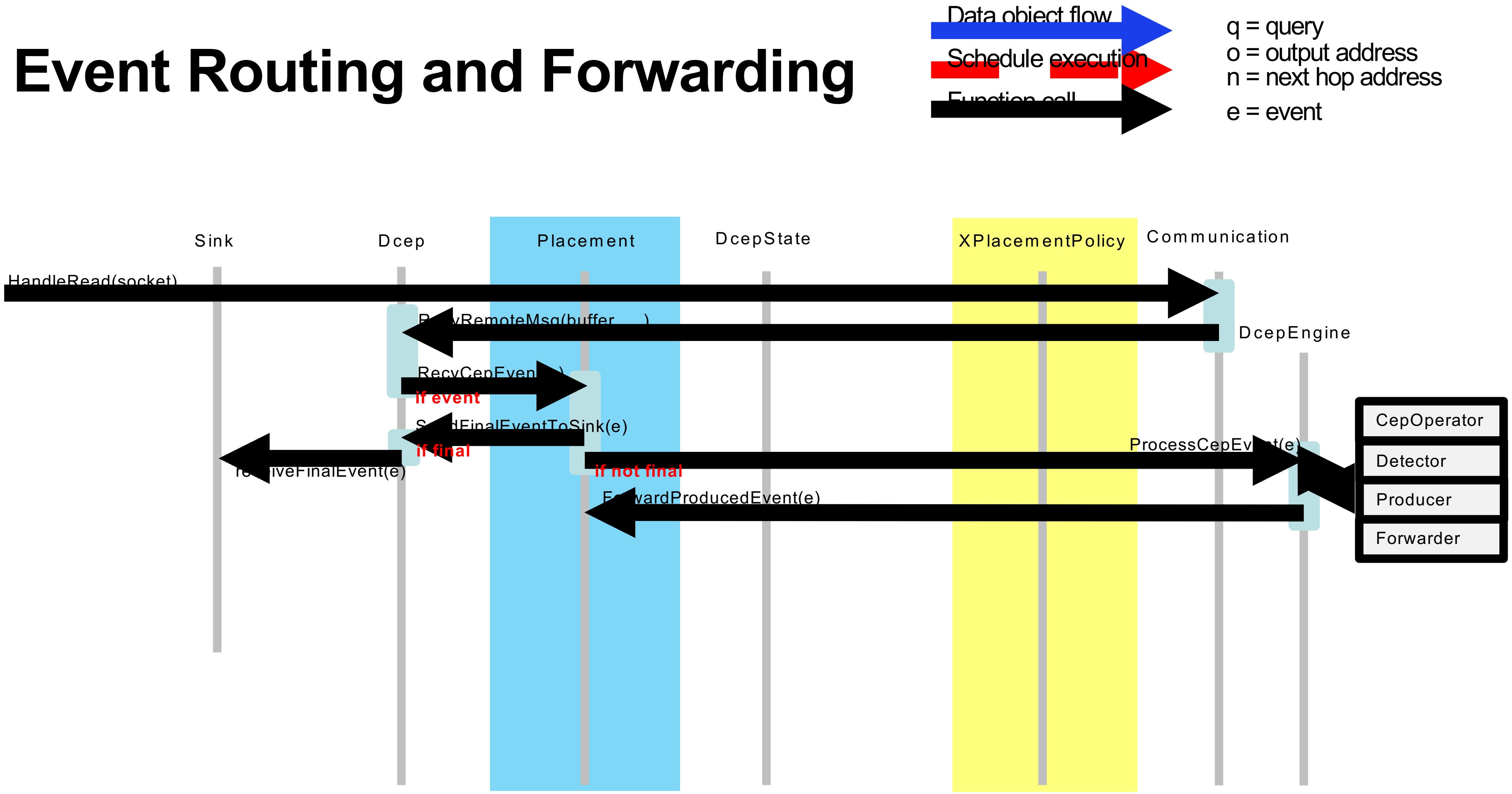

Data object flow
Schedule execution
Function call
q = query
o = output address
n = next hop address
e = event
Sink
Placement
Placement Policy
Placement Mechanism
Event routing table
CEP Engine
Communication
Sink
Dcep
Placement
DcepState
XPlacementPolicy
Communication
DcepEngine
StartApplication()
BuildAndSendQuery()
Repeated for
every operator q
DispatchQuery()
RecvQuery(q)
if query
q
q
DoPlacement()
ERT
q o,n
q
CreateEventRoutingTableEntry(q)
SetOutDest(q,o)
SetNextHop(q,n)
ForwardQuery(q)
ActivateDataSource(q)
if atomic and local
Action chosen based on decision
made by placement policy, which is
obtained from the ERT
SendPacket (q)
if remote
RecvQuery (q)
if local
ScheduleSend (q)


Data object flow
Schedule execution
Function call
q = query
o = output address
n = next hop address
e = event
Placement Node X
Placement Policy
Placement Mechanism
Communication
Network
V
A
Placement Node Z
Placement Policy
Placement Mechanism
Placement Node Z
Placement Policy
Placement Mechanism
Sink
Dcep
Placement
DcepState
XPlacementPolicy
Communication
HandleRead(socket)
DcepEngine
RecvQuery(q)
if query
q
q
DoPlacement()
ERT
q
o,n
q
ForwardQuery(q)
if atomic and local
RecvQuery (q)
if local
if remote
ScheduleSend (q)

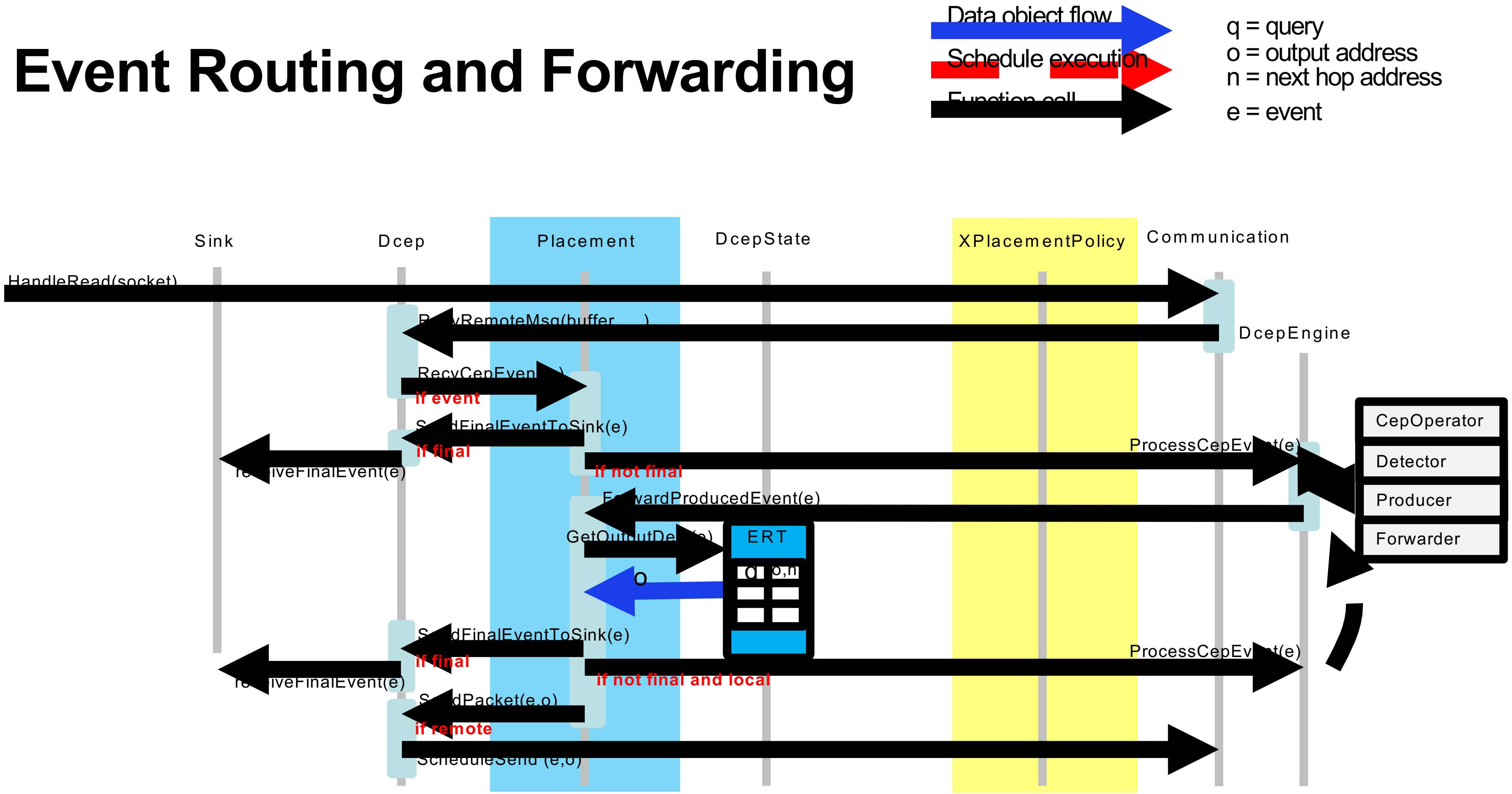
Event Routing and Forwarding



Event Routing and Forwarding



Event Routing and Forwarding



Adding a New Placement Policy

- Create a sub-class of PlacementPolicy
- Must be defined:
  - Ns-3-specific functions, attributes and trace-sources:
    - Mandatory: GetTypeID() **->**
  - configure()
    - Initialisation
  - DoPlacement()
    - Mandatory
    - Manipulate ERT via aggregated DcepState-object **->**
    - Call placement mechanism once per operator **->**

placement.cc

```
TypeId CentralizedPlacementPolicy::GetTypeId(void) {
   static TypeId tid = TypeId("ns3::CentralizedPlacementPolicy")
      .SetParent<PlacementPolicy> ()
      .AddConstructor<CentralizedPlacementPolicy> ();
   return tid;
}
```

dcep-state.h

```
class DcepState : public Object {
   …
   void SetNextHop (std::string eventType, Ipv4Address adr);
   void SetOutDest (std::string eventType, Ipv4Address adr);
```

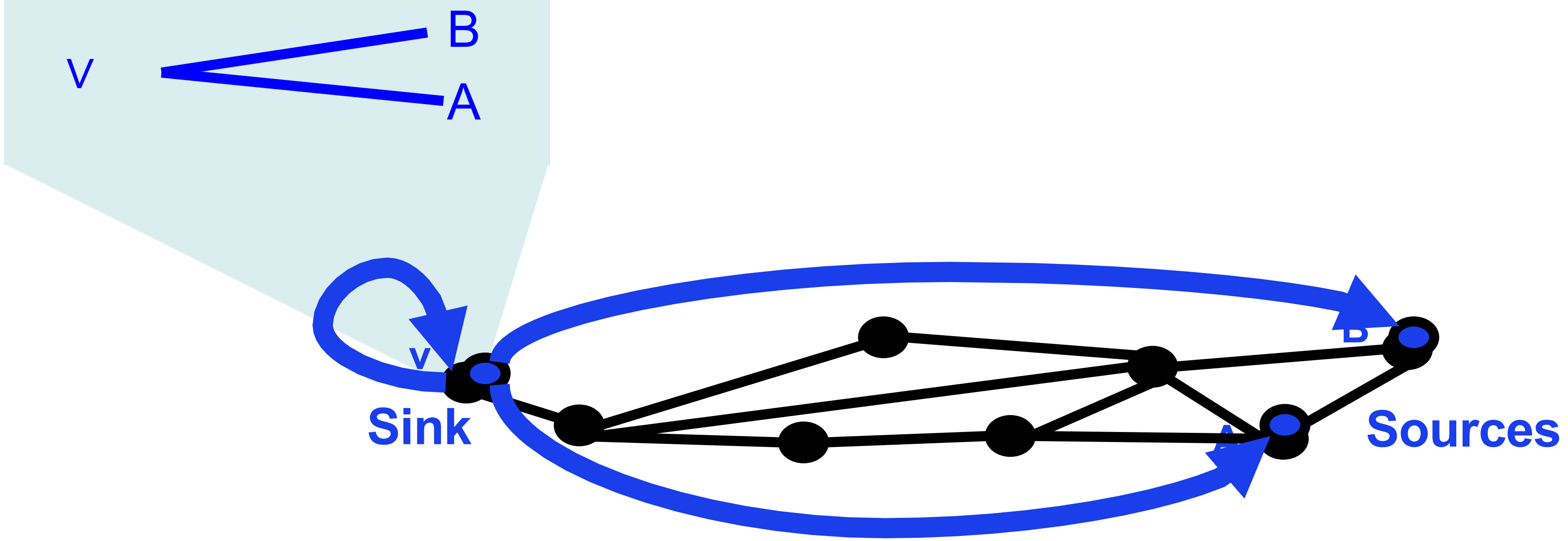
placement.h

```
class Placement : public Object {
   …
   void ForwardQuery(std::string eType);
```

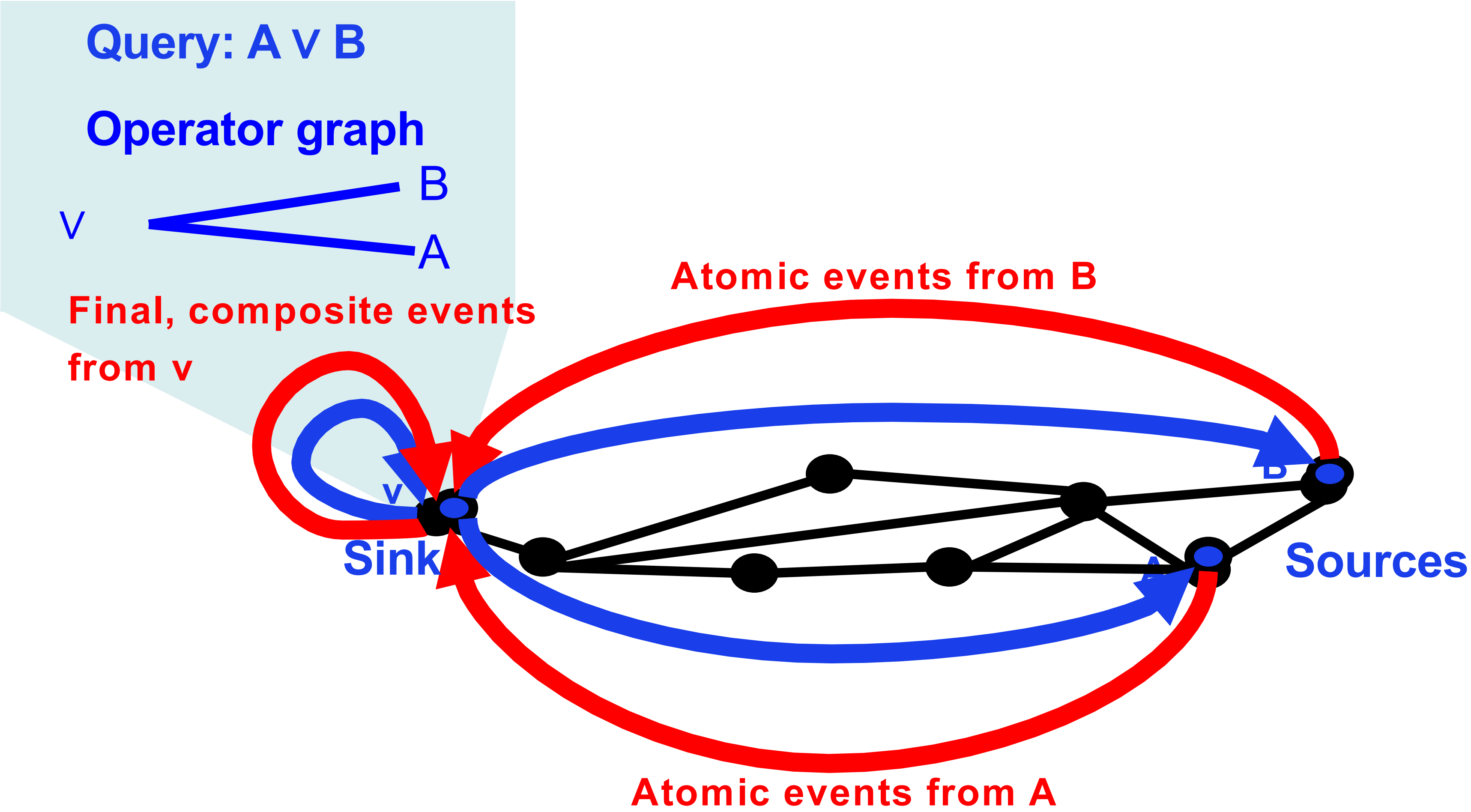
Example Placement Policy: Centralized Placement

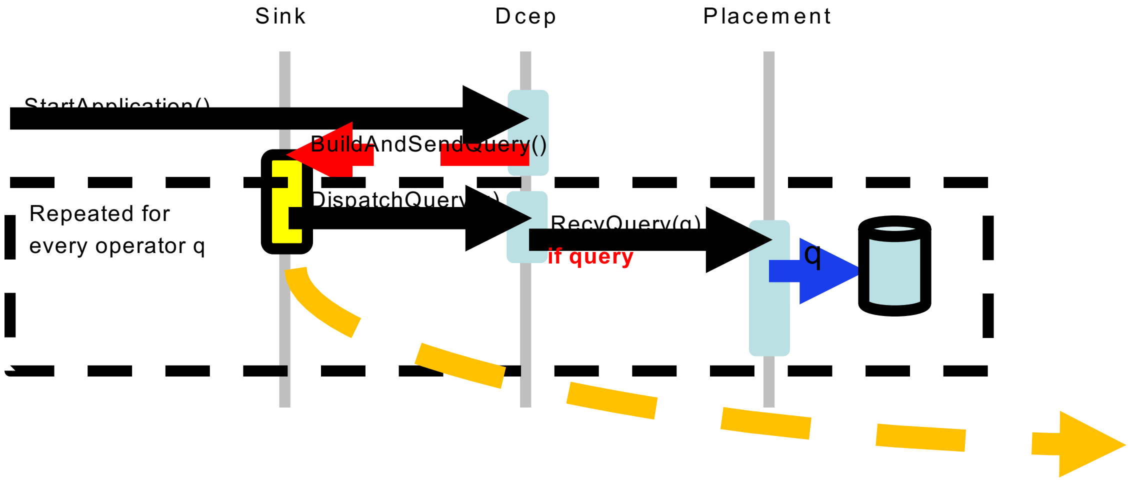
Query: A v B

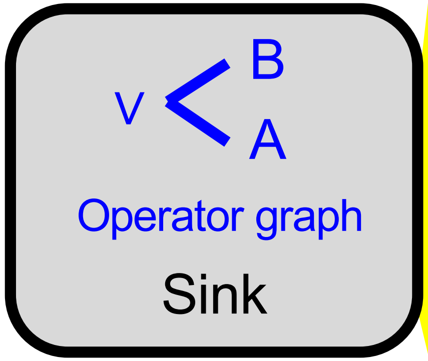
Operator graph



Example Placement Policy: Centralized Placement

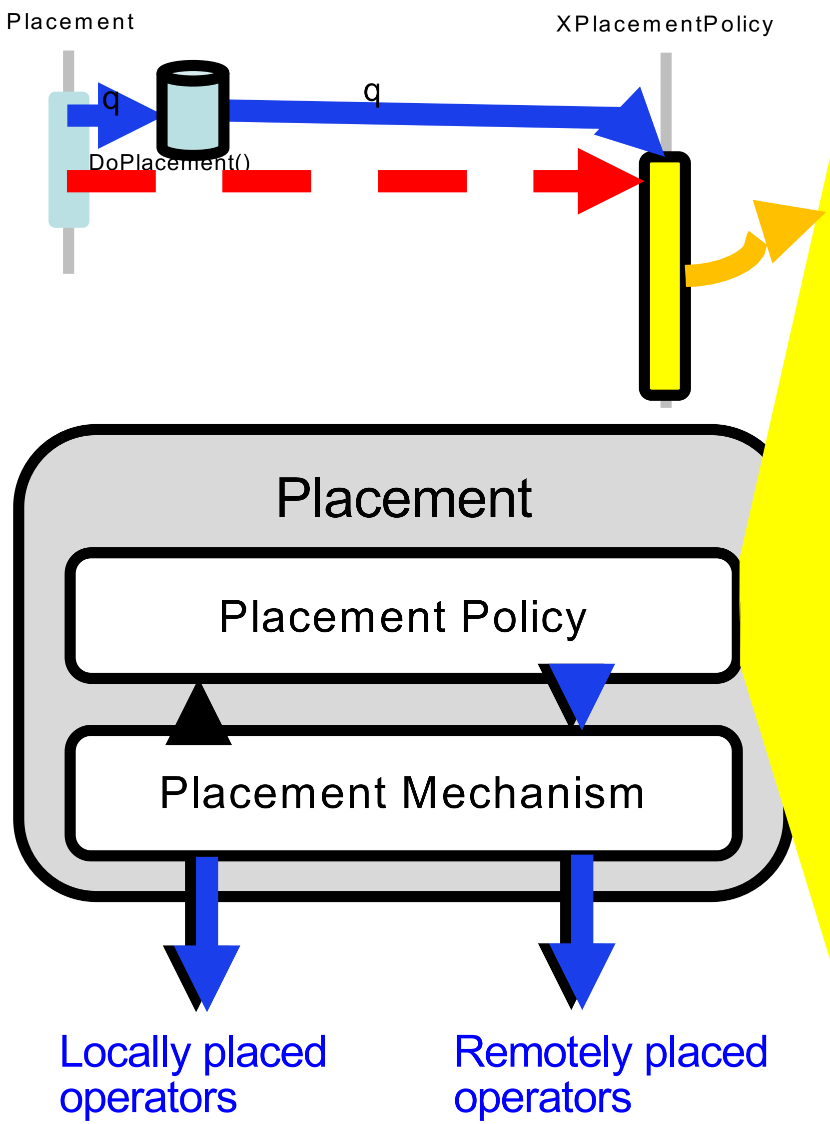






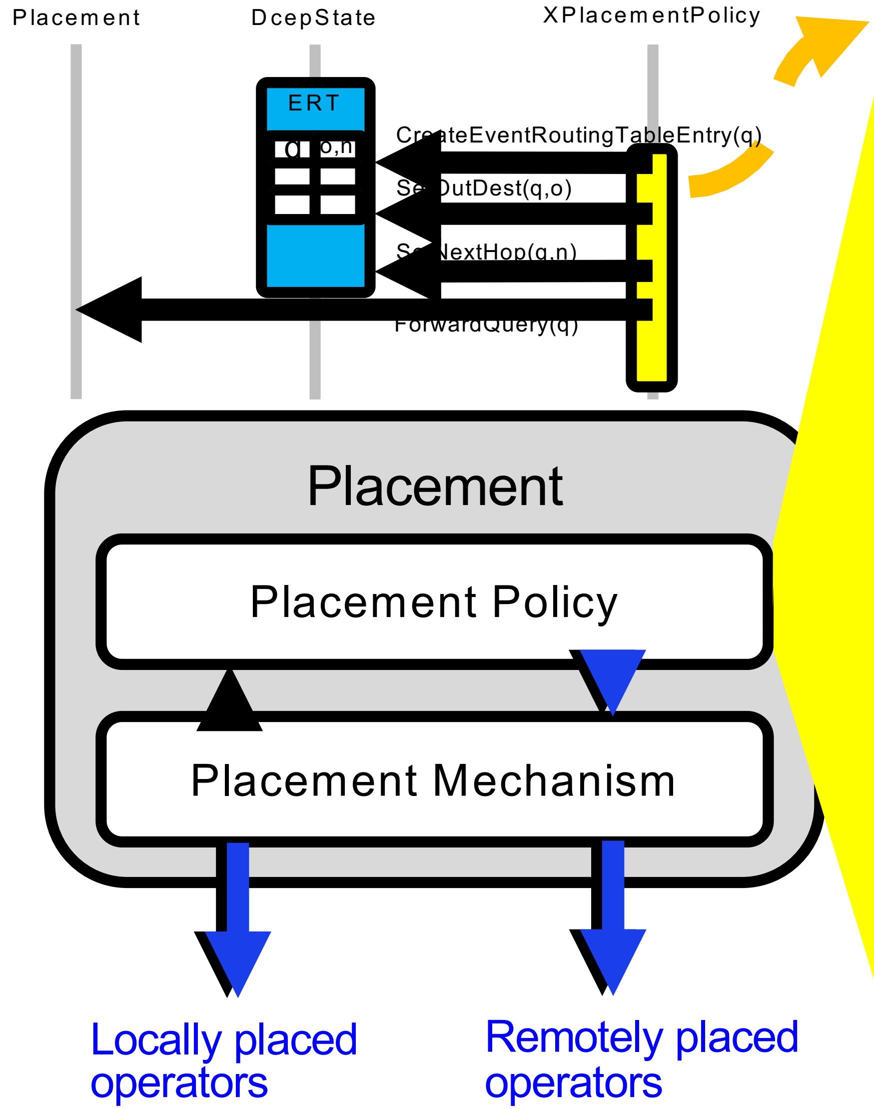
model/dcep.cc

```
void Sink::BuildAndSendQuery() {
   Ptr<Dcep> dcep = GetObject<Dcep> ();

   Ptr<Query> q1 = CreateObject<Query> ();
   q1->actionType = NOTIFICATION;
   q1->id = query_counter++;
   q1->isFinal = false;
   q1->isAtomic = true;
   q1->eventType = "A";
   q1->output_dest = Ipv4Address::GetAny();
   q1->inevent1 = "A";
   q1->inevent2 = "";
   q1->op = "true";
   q1->assigned = false;
   q1->currentHost.Set("0.0.0.0");
   q1->parent_output = "AorB";
   NS_LOG_INFO ("Setup query " << q1->eventType);
   dcep->DispatchQuery(q1);
   …
   q2->eventType = "B";
   q2->inevent1 = "B";
   dcep->DispatchQuery(q2);
   …
   q3->isFinal = true;
   q3->isAtomic = false;
   q3->eventType = "AorB";
   q3->inevent1 = "A";
   q3->inevent2 = "B";
   q3->op = "or";
   NS_LOG_INFO ("Setup query " << q3->eventType);
   dcep->DispatchQuery(q3);
}
```



model/placement.cc

```
void
  CentralizedPlacementPolicy::configure() {

  }

  void
  CentralizedPlacementPolicy::DoPlacement()
  {
     NS_LOG_INFO ("Doing centralized placement");
     Ptr<Placement> p = GetObject<Placement>();

     std::vector<Ptr < Query>>::iterator it;
     std::vector<Ptr < Query>> qs = p->q_queue;

     for (it = qs.begin(); it != qs.end(); ++it) {

        Ptr<Query> q = *it;
        if (!PlaceQuery(q))
        {
           Simulator::Schedule(Seconds(3.0),
                  &CentralizedPlacementPolicy::DoPlacement, this);
        } else
           p->RemoveQuery(q);
     }

  }
```



model/placement.cc

```
bool CentralizedPlacementPolicy::PlaceQuery(Ptr<Query> q) {
  Ptr<Placement> p = GetObject<Placement>();
  Ptr<DcepState> dstate = GetObject<DcepState>();
  dstate->CreateEventRoutingTableEntry(q);
  Ptr<Communication> cm = GetObject<Communication>();
  bool placed = false;

  if (!q->isAtomic)
     dstate->SetNextHop(q->eventType, cm->GetLocalAddress());
     placed = true;
  } else if (q->isAtomic)  {
     if(q->eventType == "A") { …
        dstate->SetNextHop(q->eventType, Ipv4Address("10.0.0.2"));
        placed = true;
     } else if(q->eventType == "B") {
        dstate->SetNextHop(q->eventType, Ipv4Address("10.0.0.3"));
        placed = true;
     } …
  }

 if (placed) {
        NS_LOG_INFO ("QUERY PLACED");
        newLocalPlacement(q->eventType);
        if(dstate->GetNextHop(q->eventType).IsEqual(cm->GetLocalAddress())) {
           NS_LOG_INFO ("QUERY PLACED ON LOCAL NODE");
           if (!q->isAtomic)
              dstate->SetOutDest(q->eventType, cm->GetLocalAddress());
           else
              dstate->SetOutDest(q->eventType, cm->GetSinkAddress());
        }
        p->ForwardQuery(q->eventType);
     }
return placed;
}
```

Add new operators

- Operator implementation based on
- CEP engine wrappers class -> detector class

As mentioned earlier:

the focus for DCEPSim until now was placement → simple event model and few operators implemented

Query vs. Operator

```
class Query : public Object
{

public:
    static TypeId GetTypeId (void);

    Query(Ptr<Query> q);
    Query();
    uint32_t id;
    uint32_t actionType;
    std::string eventType;
    bool isAtomic;
    Ipv4Address output_dest;
    Ipv4Address inputStream1_address;
    Ipv4Address inputStream2_address;
    Ipv4Address currentHost;
    std::string inevent1;
    std::string inevent2;
    std::string parent_output;
    std::string op;
    /*
     * the event notification for the event of type above is the
     * one the sink is interested in.
     */
    bool isFinal;
    bool assigned;

    SerializedQuery* serialize();
    void deserialize(uint8_t *buffer, uint32_t);
    uint32_t getSerializedSize();

};
```

Info managed during placement

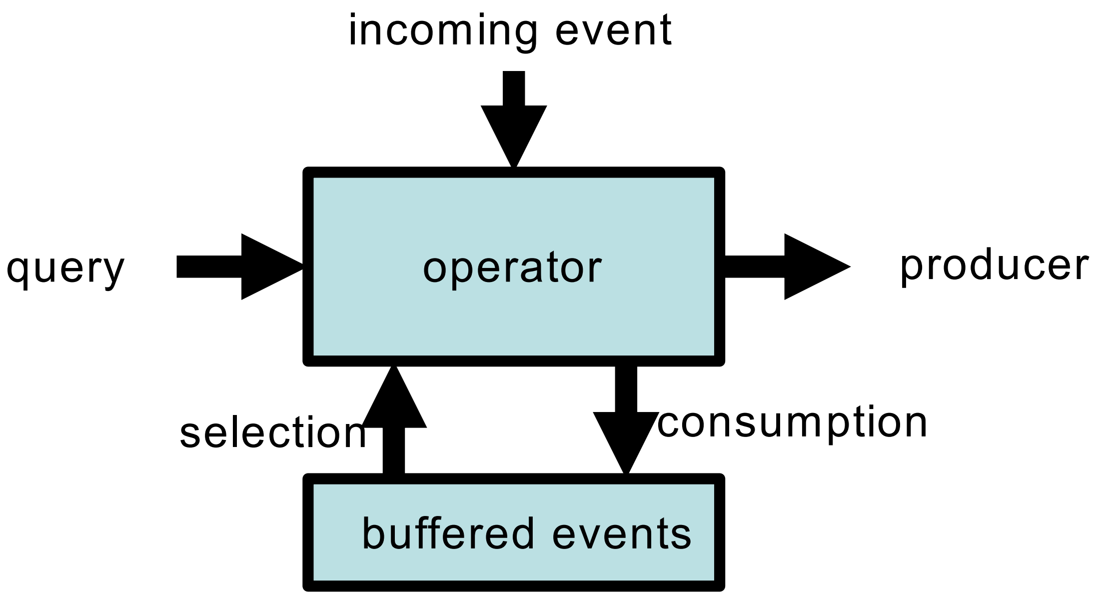
The query
- Values to be matched
- Operator

- Query used for placement
- Operator used for event processing

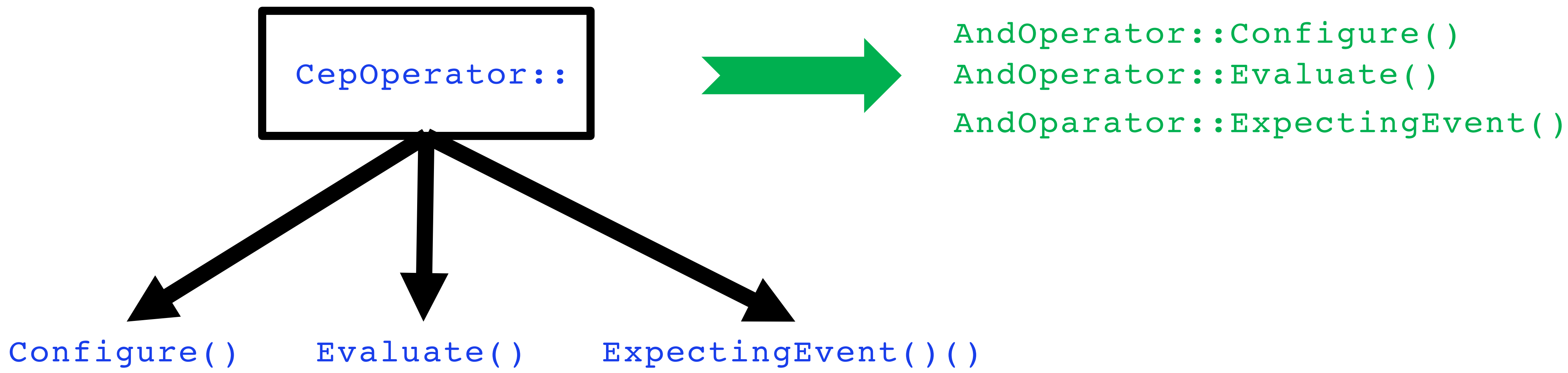
```
AndOperator::Configure(Ptr<Query> q)
{
    this->queryId = q->id;
    this->event1 = q->inevent1;
    this->event2 = q->inevent2;
```

These values are copied into an AndOperator instance

Conceptual structure of operator



Operator class in `cep-engine.cc`



Event class in `cep-engine.h`

```
 class Event : public Object{
public:
    static TypeId GetTypeId (void);

    Event(Ptr<Event>);
    Event();
    void operator=(Ptr<Event>);
    SerializedEvent* serialize();
    void deserialize(uint8_t*, uint32_t);
    uint32_t getSize();
    void CopyEvent (Ptr<Event> e);

    std::string type; //the type of the event
    uint64_t m_seq;
    uint64_t delay;
    uint32_t event_class;
    int32_t hopsCount;
    int32_t prevHopsCount;
};
```

```
void
AndOperator::Configure(Ptr<Query> q)
{
    this->queryId = q->id;
    this->event1 = q->inevent1;
    this->event2 = q->inevent2;

    Ptr<BufferManager> bufman = CreateObject<BufferManager>();

    bufman->consumption_policy = SELECTED_CONSUMPTION; //default
    bufman->selection_policy = SINGLE_SELECTION; //default
    bufman->configure(this);
    this->bufman = bufman;
}
```

Copy info from query object during placement

Create a buffer manager for the operator

Set consumption and selection policies

```
AndOperator::Evaluate(Ptr<Event> e, std::vector<Ptr<Event> >& returned)
{
    std::vector<Ptr<Event>> events1;
    std::vector<Ptr<Event>> events2;
    bufman->read_events(events1, events2);

    if((!events1.empty()) && (!events2.empty()))
    {
      if (e->type == events1.front()->type)
      {
          std::vector<Ptr<Event>>::iterator it = events2.begin();
          for (uint32_t i = 0; i < events2.size(); i++, it++)
          {

            if(e->m_seq == bufman->events2[i]->m_seq)
            {
                Ptr<Event> e1 = CreateObject<Event>();
                Ptr<Event> e2 = CreateObject<Event>();
                e->CopyEvent(e1);
                events2[i]->CopyEvent(e2);

                bufman->events2.erase(it);
                returned.push_back(e1);
                returned.push_back(e2);



                return true;
            }
          }

```

Make sure both buffers are not empty

Check which buffer the event belongs to

Find the event from the other buffer which matches the sequence number of the current event

```
bool
AndOperator::ExpectingEvent(std::string eType)
{
    if((event1 == eType) || (event2 == eType))
        return true;
    else
        return false;
}
```

Determines whether this operator is expecting events of the type provided as parameter.

Conclusions

- DCEPSim is
  - a tool for our research in operator placement for mobile distributed CEP
  - not perfect
  - but «easily» extensible (especially if one gets aquainted with ns-3)

- In case you have any questions/ideas/comments
  - Talk with us here @ DEBS 2018
  - Email us: fabriceb@ifi.uio.no, steikr@ifi.uio.no, plageman@ifi.uio.no