# Computer Supported Collaborative Knowledge building : P2P Semantic Wikis Approach


Hala Skaf-Molli

ECOO Team

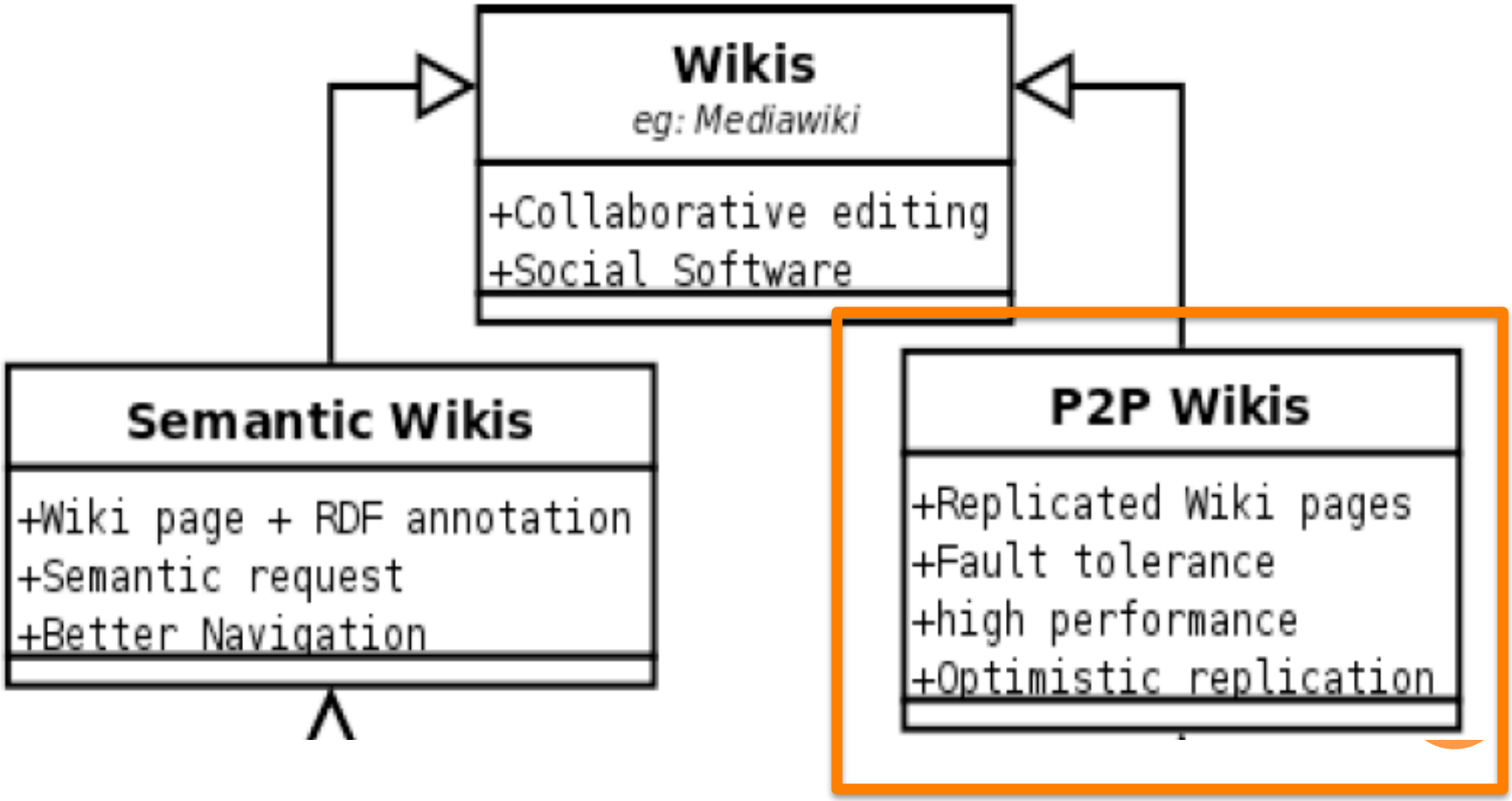
Associate Professor

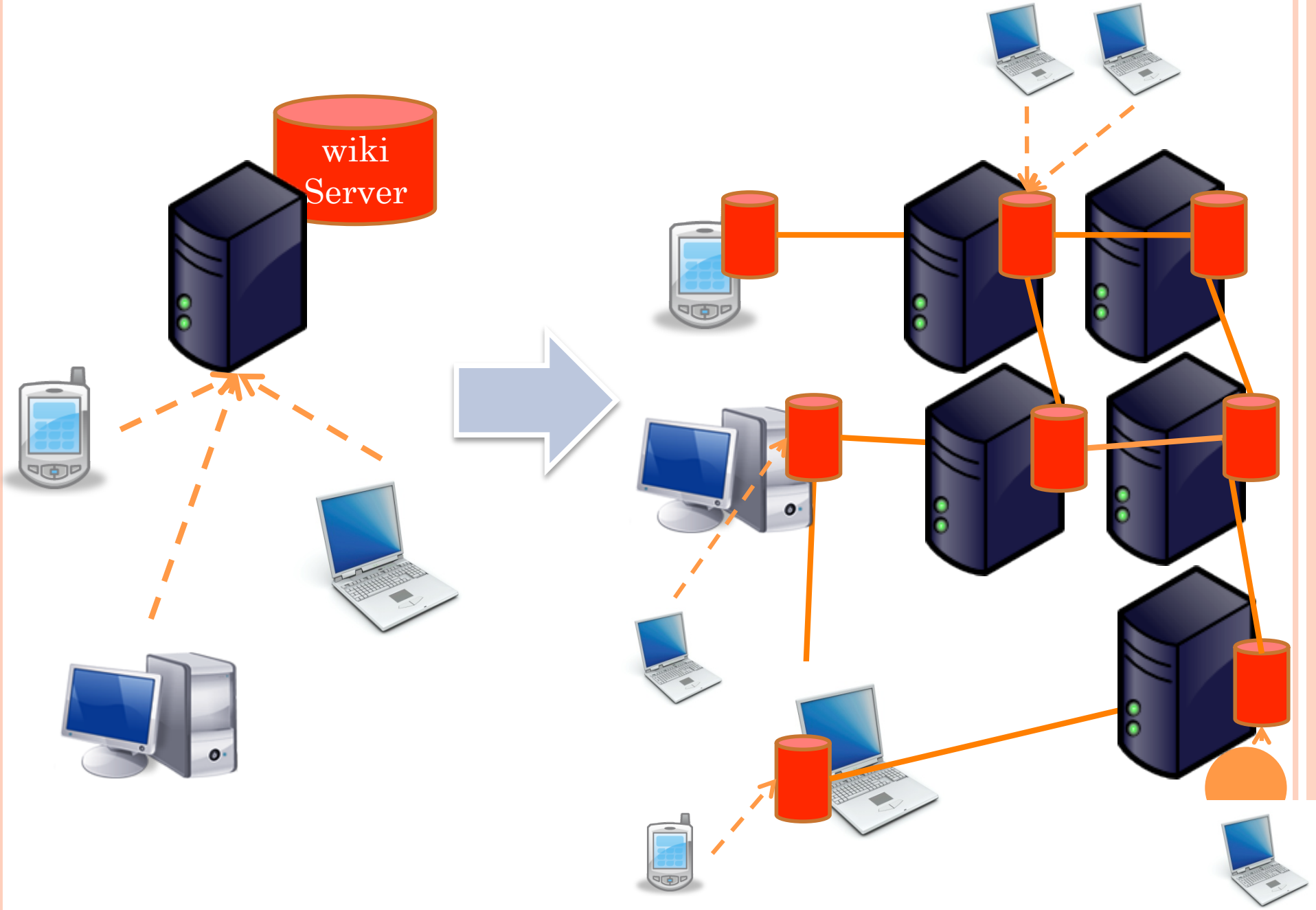
Nancy-University

skaf@loria.fr

http://www.loria.fr/~skaf

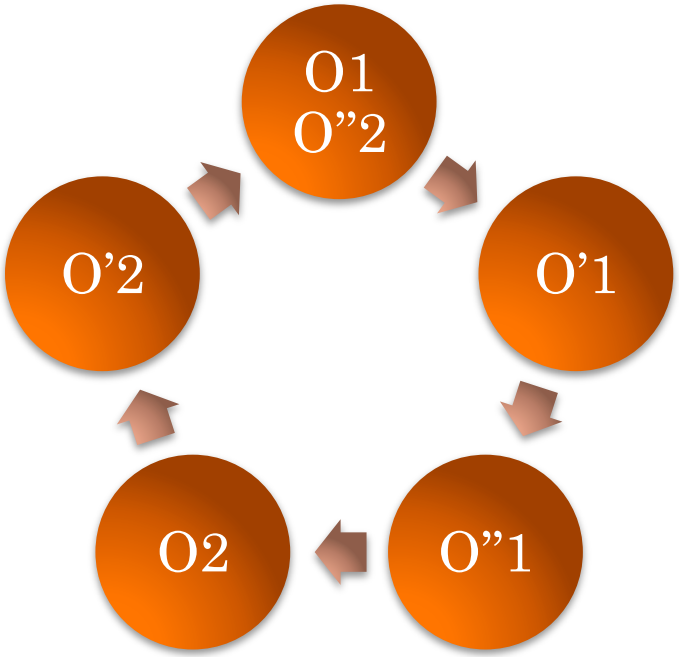
# Wikis Evolutions



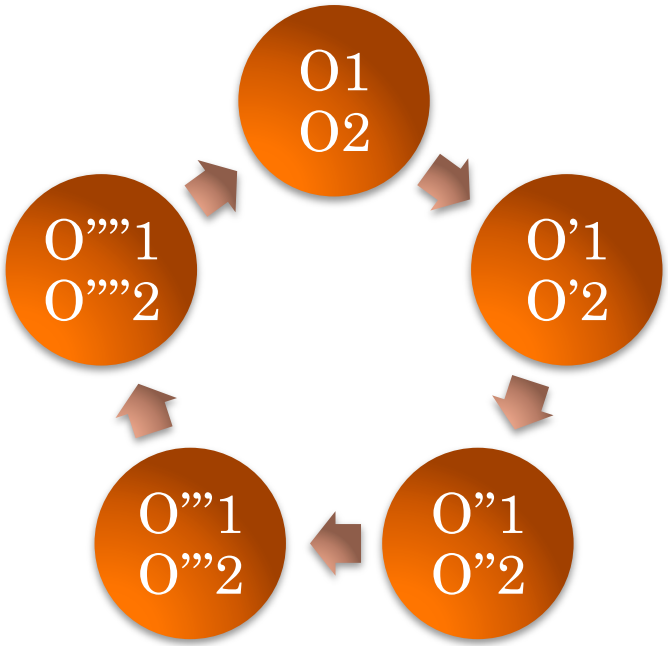

wiki
Server

# Peer to peer Wikis

- Every peer  acts as a client and a server
- Wikis pages are replicated on the peers
  - Partially or totally




O1 has 3 replicas,
O2 has 3 replicas




Number of replicas =  number of peers

# P2P Wikiks

- Advantages
  - Better performance and availability
  - Fault-tolerance
  - Load-balancing
- In all cases (partial or total) rely on **optimistic replication approach**

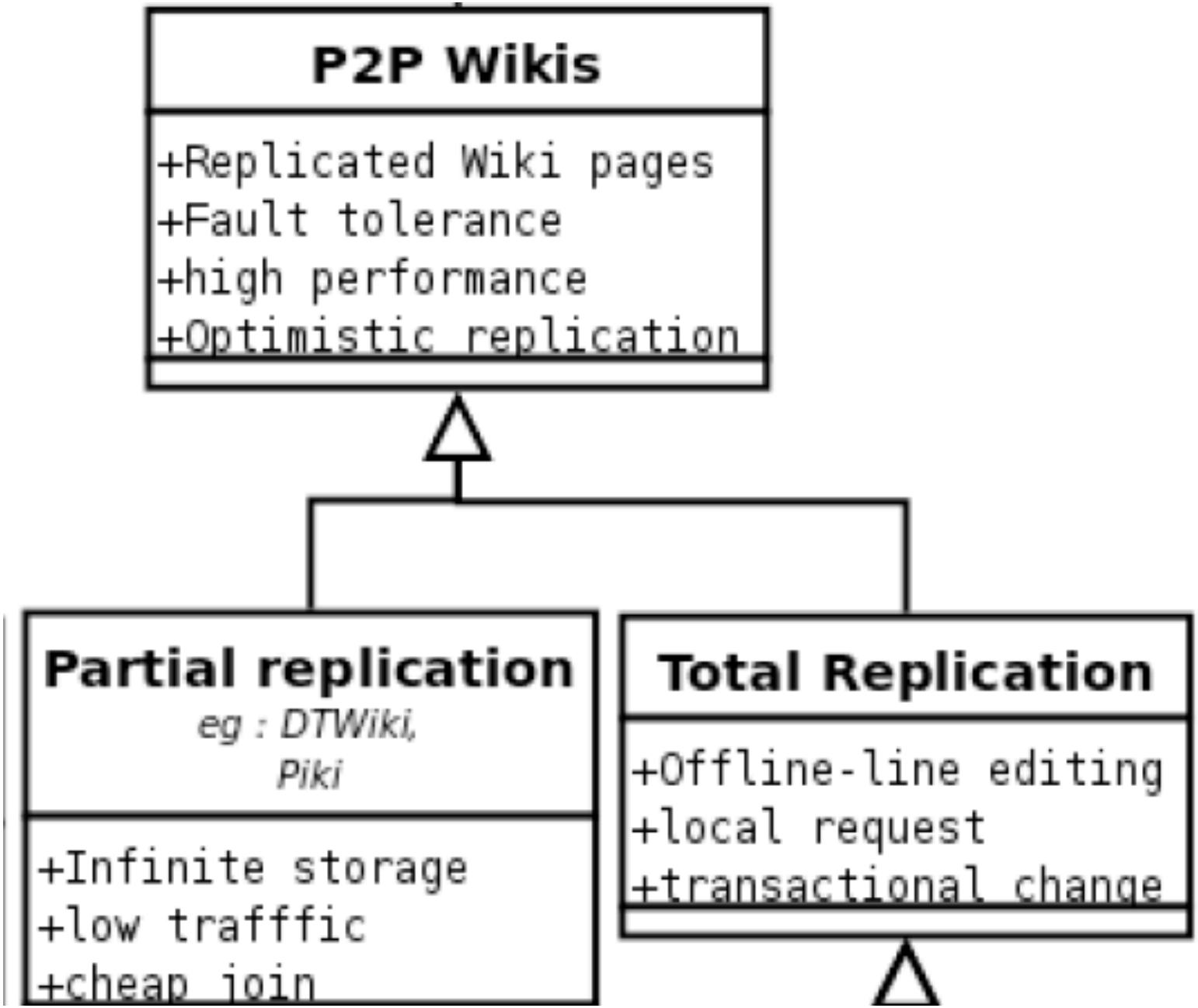
# General model of optimistic replication Approach

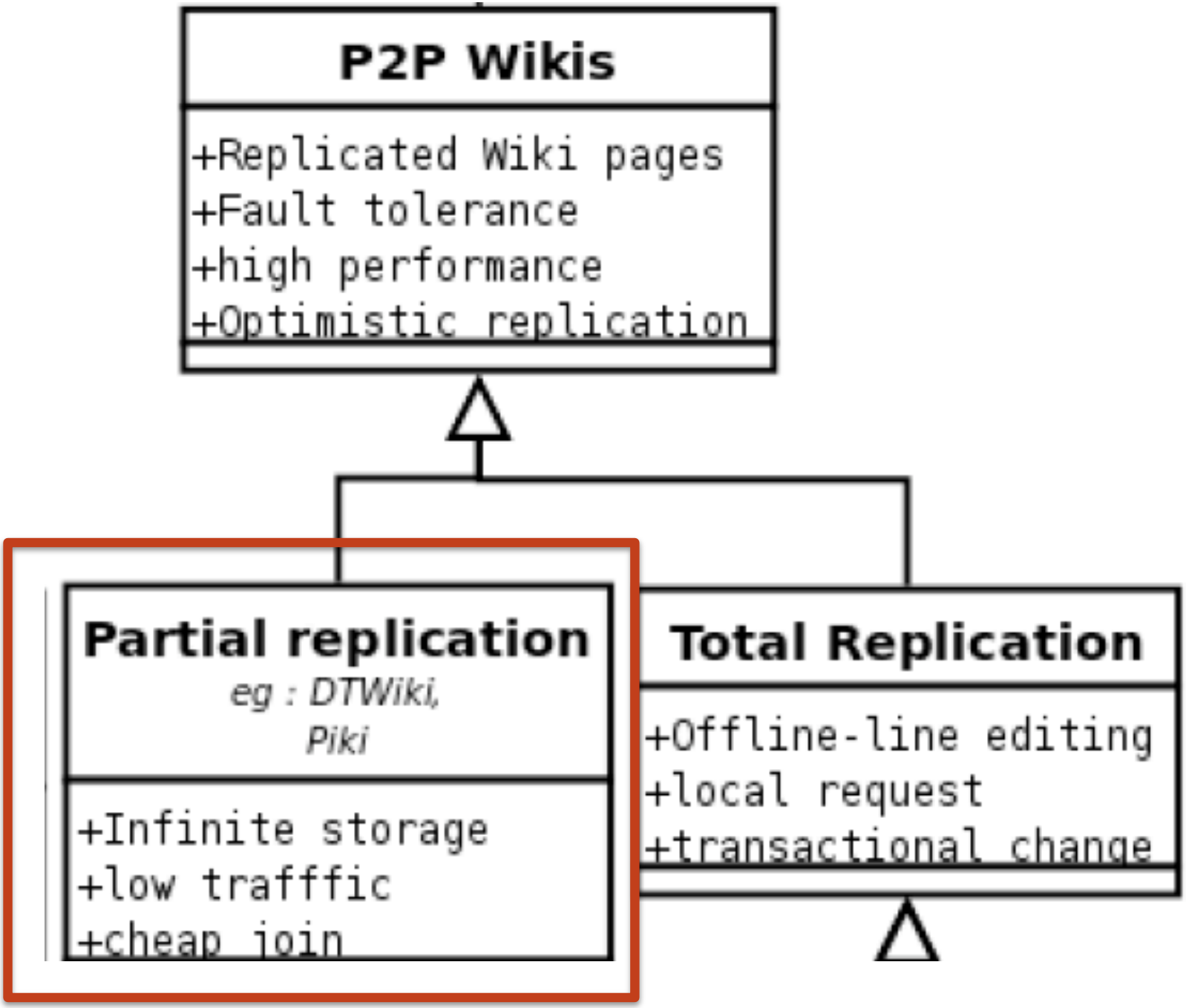
- An operation is generated on  one site and executed immediately
  - without any locking, any communication with another site
- Operation is broadcasted to other sites
  - Hypothesis: the operation will eventually arrive on all sites (not so easy…)
- When a site receives a remote operation
  - The operation is re-executed
- The system is correct if it ensures a consistency criteria such as Causal Consistency, Eventual Consistency, CCI consistency etc…

# Consistency Levels

- **Causal Consistency** : dependent operations are executed in the same order in all servers
- **Eventual consistency**: all copies are identical when the system is idle
- **CCI consistency:** Causal consistency + Eventual consistency + Intention preservation
  - **Intention preservation:** The effect observed at generation time is observed on all sites whatever any concurrent operation.

Sun and et al. Achieving Convergence, Causality Preservation, and Intention Preservation in Real-Time Cooperative Editing Systems, *ACM Transactions on Computer-Human Interaction*, 5(1), 1998.


P2P Wikis
+Replicated Wiki pages
+Fault tolerance
+high performance
+Optimistic replication
Partial replication
eg : DTWiki,
Piki
+Infinite storage
+low trafffic
+cheap join
Total Replication
+Offline-line editing
+local request
+transactional change


P2P Wikis
+Replicated Wiki pages
+Fault tolerance
+high performance
+Optimistic replication
Partial replication
eg : DTWiki,
Piki
+Infinite storage
+low trafffic
+cheap join
Total Replication
+Offline-line editing
+local request
+transactional change

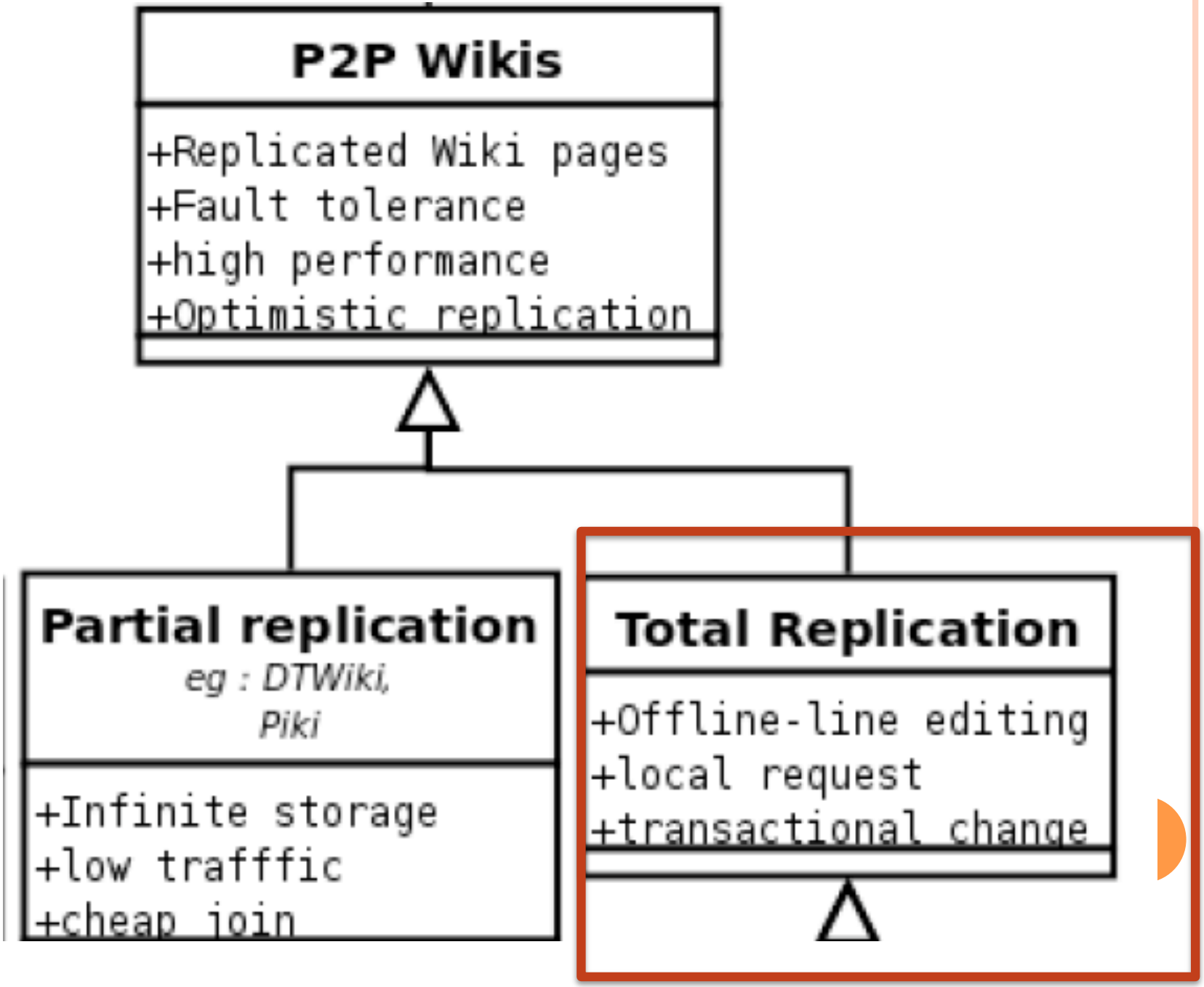
# Partialy replicated P2P Wikis

- Based on Structured P2P network
  - Distributed Hash Table (DHT) networks
  - Efficient resource discovery
- Wiki pages are replicated on a number of *replicas peers*
- A limit number of replicas of every page.
- Examples:
  - DistiWiki, DTWiki, Piki

Morris. DistriWiki: a distributed peer-to-peer wiki network, international symposium on Wikis (WikiSym), 2007.
Du et al. DTWiki: a disconnection and intermittency tolerant wiki, 17th international conference on World Wide Web (WWW), 2008.
Mukherjee et al. Piki - A Peer-to-Peer based Wiki Engine. Eighth International Conference on Peer-to-Peer Computing (P2P08), 2008.

# Partialy Replicated P2P Wikis

- Advantages:
  - Infinite storage capacity
  - Low traffic : operation on a page is just propagated to few peer, not at all peers…
  - Joining the P2P network is cheap : may require to transfer few pages on join
- Drawbacks:
  - Do not ensure the CCI correction model (DistriWiki,Piki, DTWiki)
    - User-based merge and manual resolution of conflicts
  - Cannot process a request locally
  - Do not support offline work
  - Do not enable transactional changes


P2P Wikis
+Replicated Wiki pages
+Fault tolerance
+high performance
+Optimistic replication
Partial replication
eg : DTWiki,
Piki
+Infinite storage
+low trafffic
+cheap join
Total Replication
+Offline-line editing
+local request
+transactional change

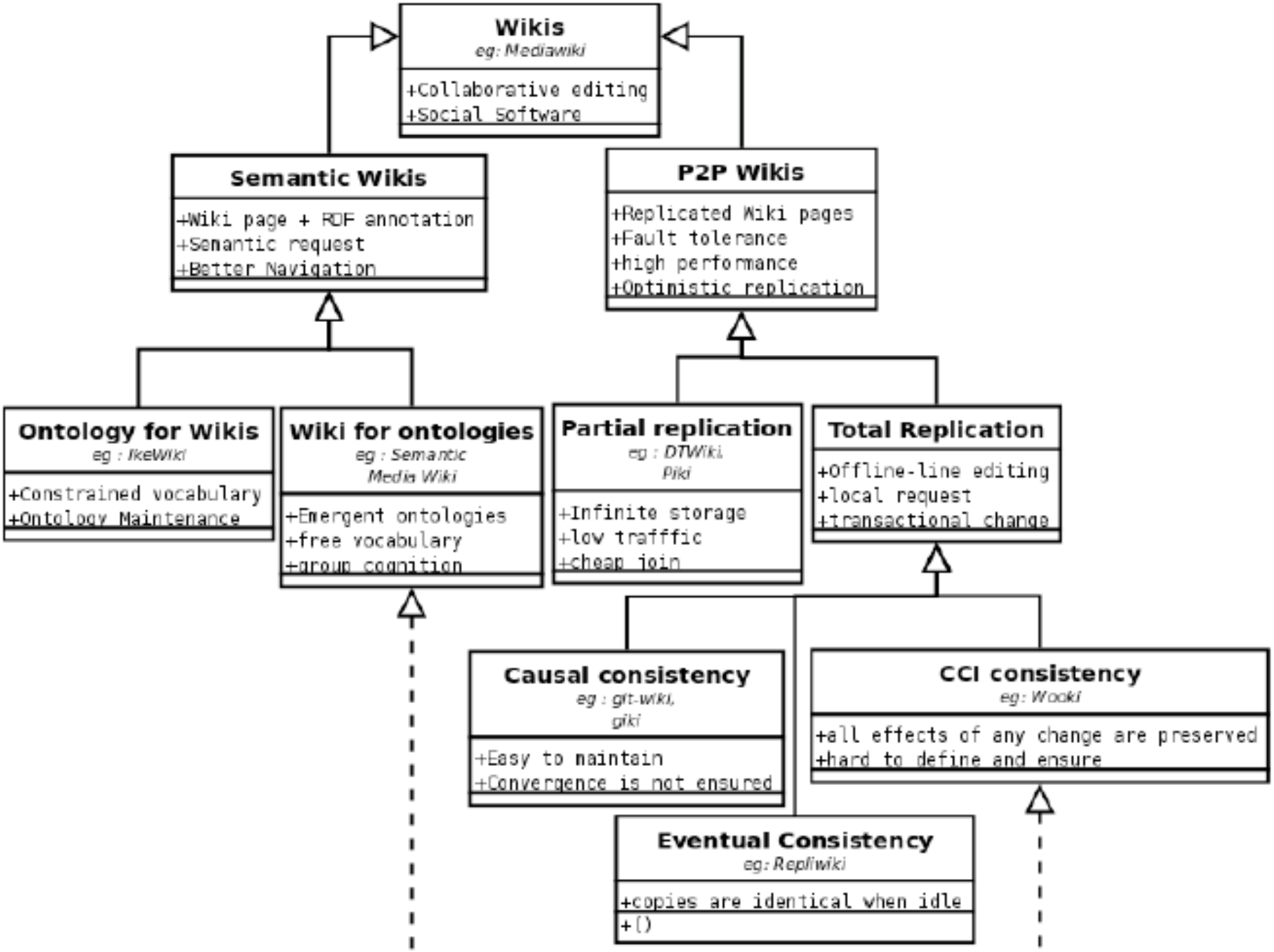
# Totally Replicated P2P wikis

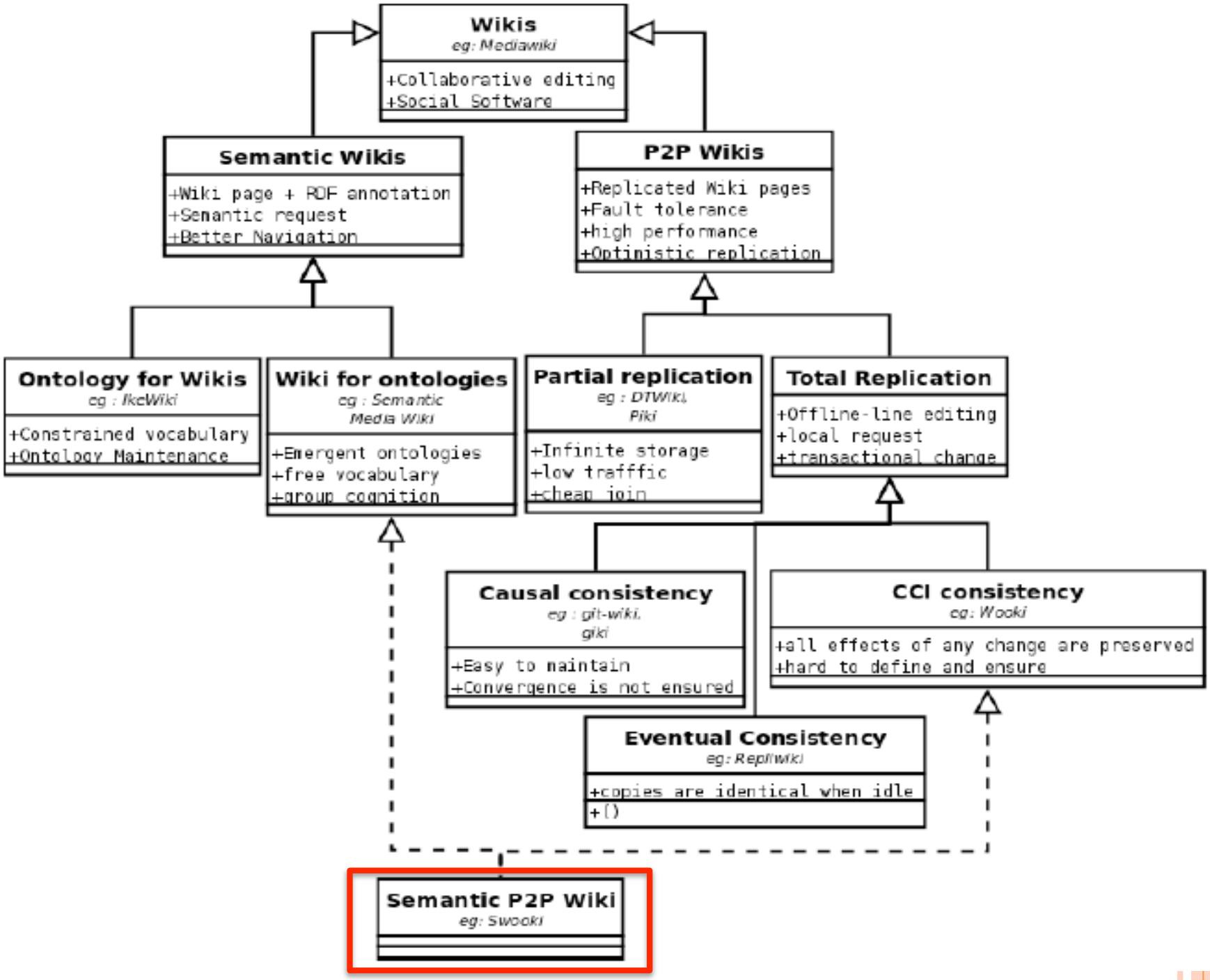
- Each wiki server hosts a copy of the set of wiki pages.
- Number of replicas is equal to the number of the servers
- Examples:
  - Git-wiki (http://atonie.org/2008/02/git-wiki)
    - Causal Consistency
  - RepliWiki:
    - Eventual Consistency
  - Wooki:
    - CCI Consistency (Woot Algorithm)

Weiss et al. Wooki: a P2P Wiki-based Collaborative Writing Tool, Web Information Systems Engineering (WISE), 2007

# Totally Replicated P2P Wikis

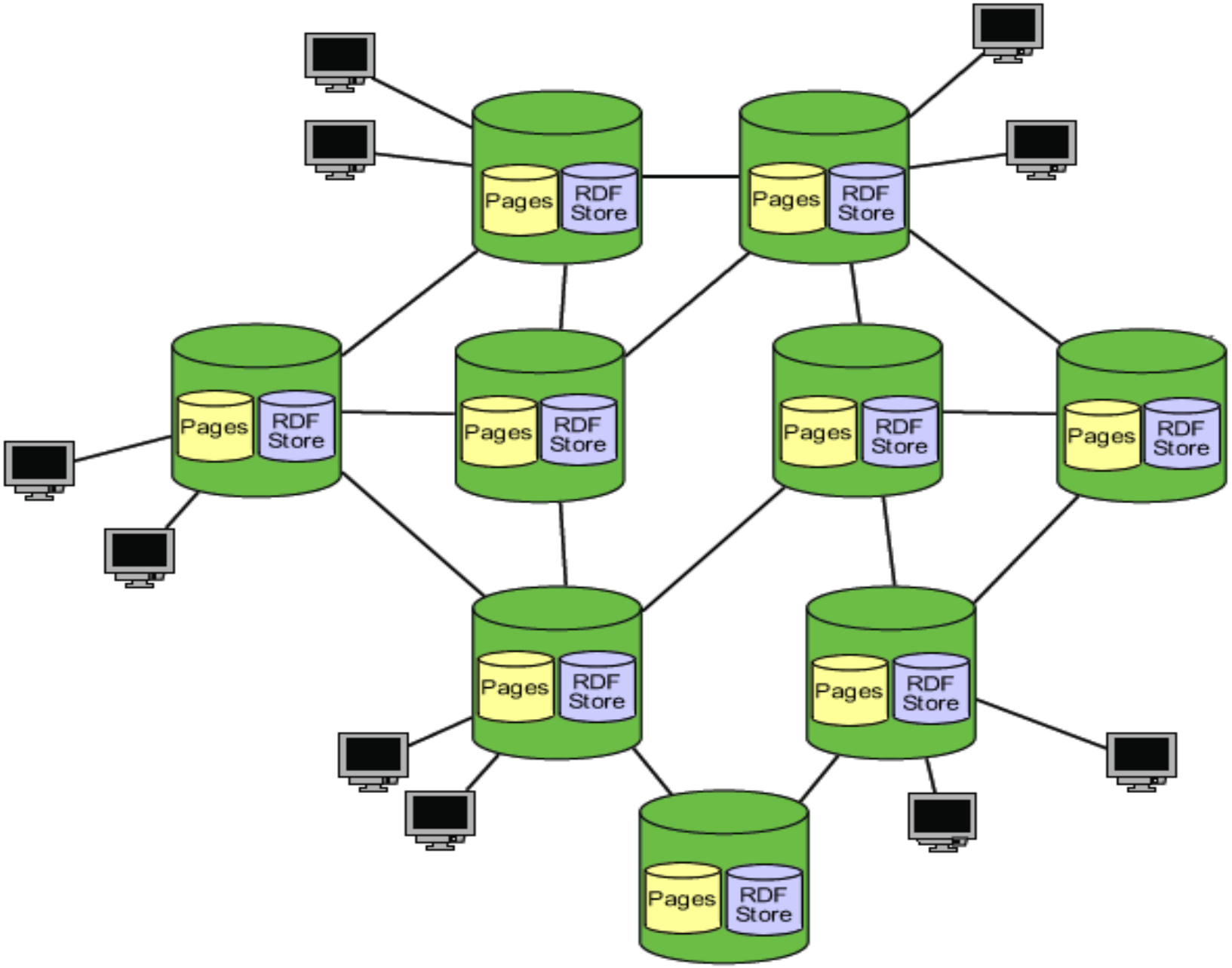
- Advantages:
  - Scalability with user number
  - WOOKI ensures CCI consistency
  - Local queries
  - Support offline work
  - Enable transactional changes
- Drawbacks:
  - Joining the P2P network is expensive : state transfer
  - High traffic : operation on a page is propagated all peers…


Wikis
eg: Mediawiki
+Collaborative editing
+Social Software
Semantic Wikis
+Wiki page + RDF annotation
+Semantic request
+Better Navigation
P2P Wikis
+Replicated Wiki pages
+Fault tolerance
+high performance
+Optinistic replication
Ontology for Wikis
eg : IkeWiki
+Constrained vocabulary
+Ontology Maintenance
Wiki for ontologies
eg : Semantic Media Wiki
+Emergent ontologies
+free vocabulary
+group cognition
Partial replication
eg : DTWiki, Piki
+Infinite storage
+lov trafffic
+cheap join
Total Replication
+Offline-line editing
+local request
+transactional change
Causal consistency
eg : git-wiki, giki
+Easy to maintain
+Convergence is not ensured
CCI consistency
eg: Wooki
+all effects of any change are preserved
+hard to define and ensure
Eventual Consistency
eg: Repliwiki
+copies are identical when idle
+()


Wikis
eg: Mediawiki
+Collaborative editing
+Social Software
Semantic Wikis
+Wiki page + RDF annotation
+Semantic request
+Better Navigation
P2P Wikis
+Replicated Wiki pages
+Fault tolerance
+high performance
+Optimistic replication
Ontology for Wikis
eg : IkeWiki
+Constrained vocabulary
+Ontology Maintenance
Wiki for ontologies
eg : Semantic Media Wiki
+Emergent ontologies
+free vocabulary
+group cognition
Partial replication
eg : DTWiki, Piki
+Infinite storage
+low trafffic
+cheap join
Total Replication
+Offline-line editing
+local request
+transactional change
Causal consistency
eg : git-wiki, giki
+Easy to maintain
+Convergence is not ensured
CCI consistency
eg: Wooki
+all effects of any change are preserved
+hard to define and ensure
Eventual Consistency
eg: Repliwiki
+copies are identical when idle
+()
Semantic P2P Wiki
eg: Swooki

# SWOOKI: P2P SEMANTIC WIKIS

- Integrate advantages of :
  - P2P Wikis based on total replication
  - With CCI consistency
- And
  - Semantic wiki based on the approach “wiki for ontologies”


Pages
RDF Store
Pages
RDF Store
Pages
RDF Store
Pages
RDF Store
Pages
RDF Store
Pages
RDF Store
Pages
RDF Store
Pages
RDF Store
Pages
RDF Store

# SWOOKI

- Why wiki for ontologies:
  - We are interested in ontology emergence
- Why total replication ?
  - Queries can be executed locally
  - Offline work
  - Transactional changes i.e.
- Modifying a semantic wiki like Semantic Media Wiki requires to change many pages (one subject per page) If no transactional change :
  - Visibility of intermediate result is confusing for other users
  - Requests view intermediate results

# Challenges

- New data Type
  - Text and semantic data
- Editing operations
- Optimistic replication algorithm
  - Adapted to the new data model
  - Adapted to P2P constraints
  - Ensures CCI Consistency

# Data Model

- A wiki Page is a sequence of lines
- Lines contain text and semantic data

France is located in [located In::Europe]

The capital of France is [has Capital::Paris]

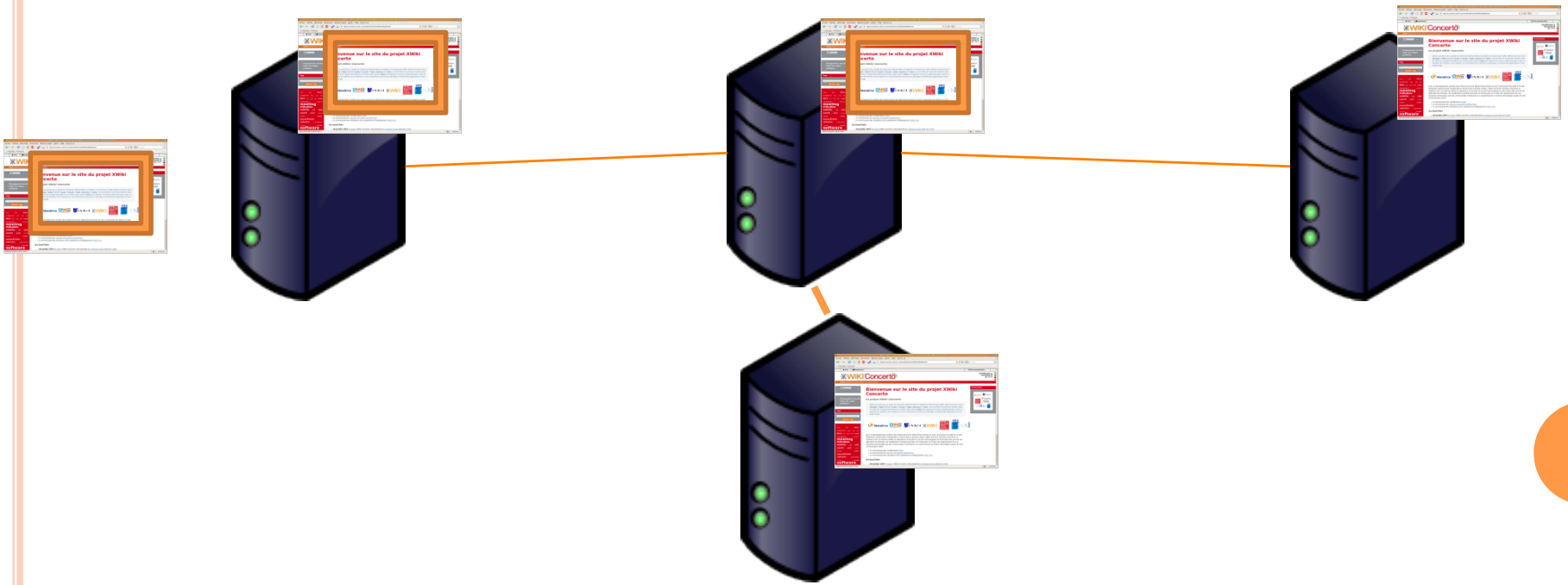
- Two editing operations
  - Insert a new line
  - Delete an existing line
  - Update = Delete + Insert

# CCI Model for SWOOKI

- Convergence for text (by WOOT) and semantic data (??)
- Causality same as  WOOKI
  - SWOOKI do not introduce new editing operations
  - Pre-conditions on editing operation
- Intention preservation:
  - Intention: text (insert (lp < l < ln)), semantic data (?)
  - Preservation: text (WOOT), semantic data(?)

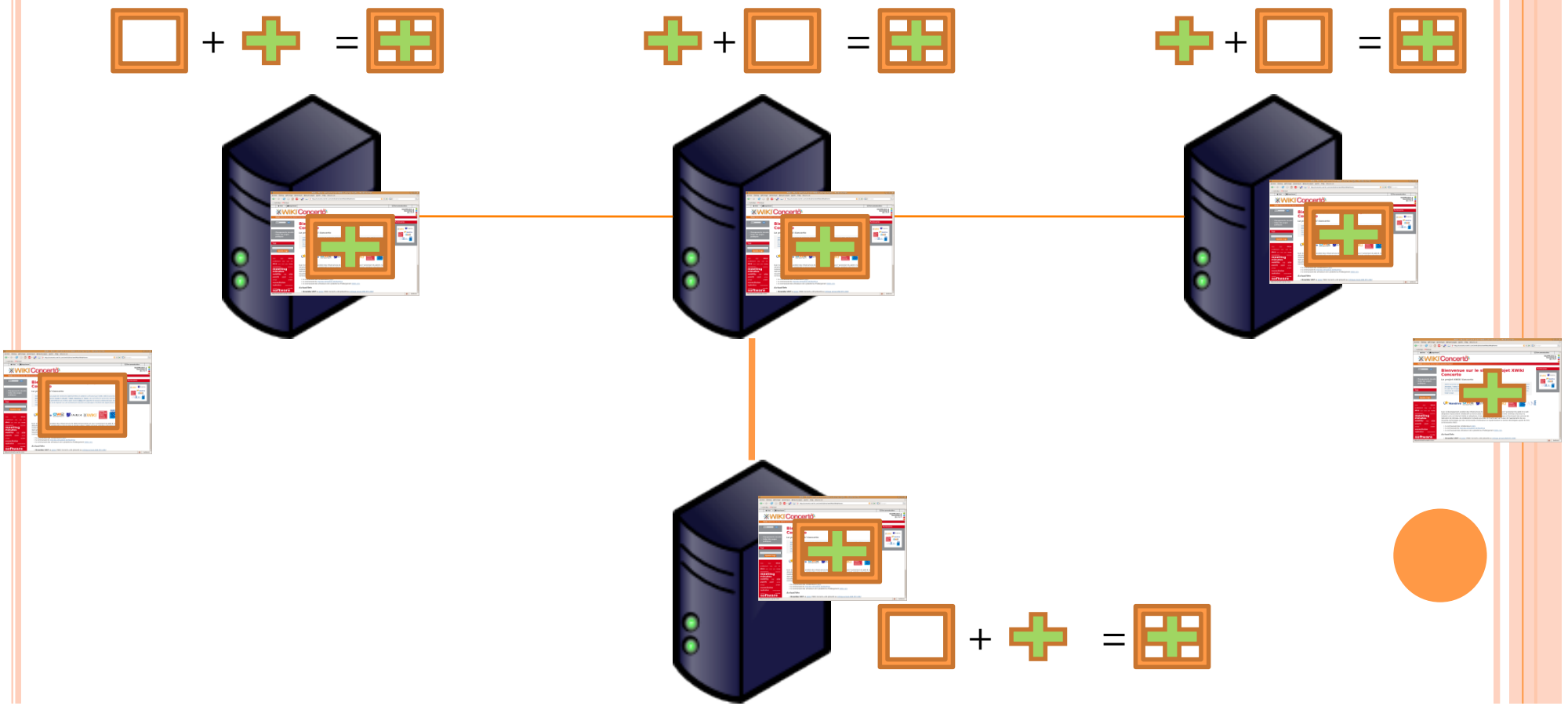
# WOOT Approach

- A change is immediatly applied, broadcast to others (epidemic), integrated by remote (optimistic replication).



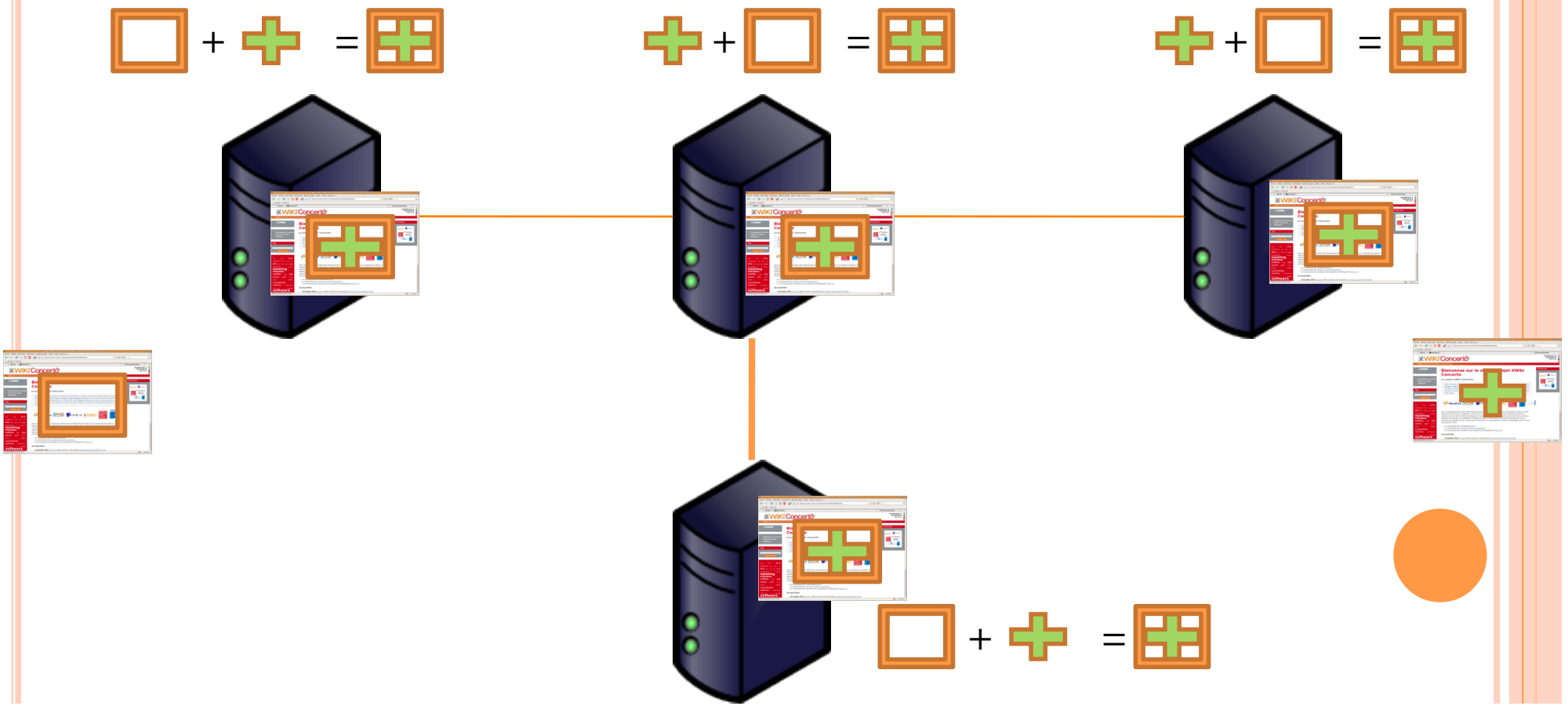
# Optimistic replication and concurrent changes

- Automatic merge must be executed by servers when receiving remote changes…



# Optimistic replication and concurrent changes

- Automatic merge must be executed by servers when receiving remote changes…



# Optimistic replication for RDF Model

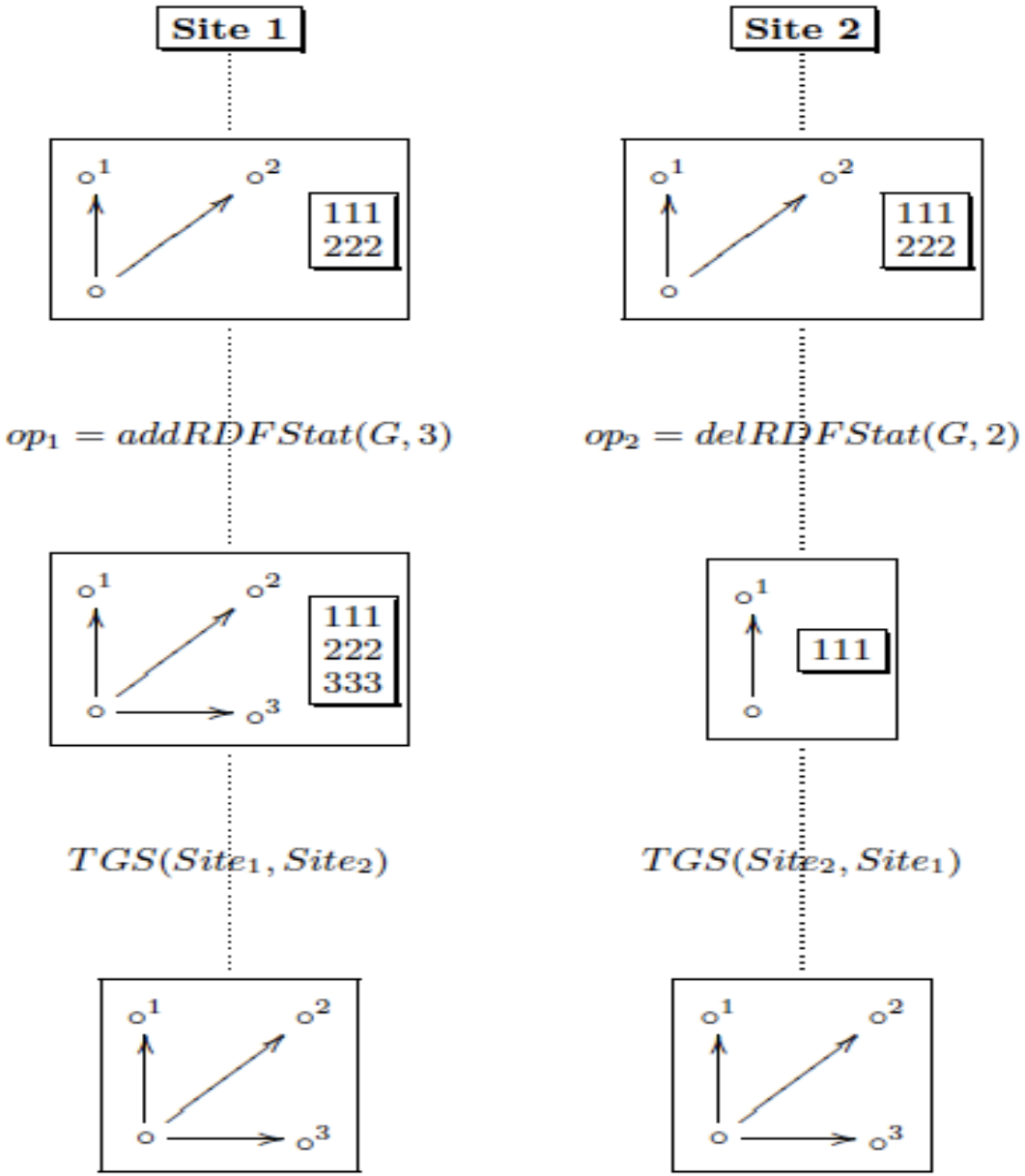
- RDFPeers
  - Partial replication to ensure fault tolerance
  - Each RDF Triple is stored in 3 places
- Problems:
  - Sharing immutable semantic data
  - Concurrent updates are not managed
- RDFGrowth and Publish/Subscribe Networks

Cai et al. RDFPeers: A Scalable Distributed RDF Repository based on a Structured Peer-to-Peer Network. 13 international conference on World Wide Web (WWW), 2004.
Chirita et al. Publish/Subscribe for RDF-based P2P Networks. The Semantic Web: Research and Applications, First European Semantic Web Symposium, ESWS 2004, LNCS 3053

# Optimistic replication for RDF Model

- RDFSync: Synchronize a source and a target of RDF data
  - Decompose RDF graphs into Minimal Self Contained Graphs (MSGs)
  - Ordered list of identifiers (hashs)
- Convergence but does not preserve intention on delete

Morbidoni et al. Rdfsync: Efficient Remote Synchronisation of RDF models. 6th International Semantic Web Conference, ISWC, 2007.




Convergence but does not preserve intention on delete

# Optimistic replication for RDF Model

- Edutella RDF-based metadata infrastructure for P2P applications.
  - Querying distributed RDF metadata
  - Replication service :
    - metadata persistence / availability
    - workload balancing
    - maintaining metadata integrity and consistency
  - Do not mention how to replicate and synchronize metadata

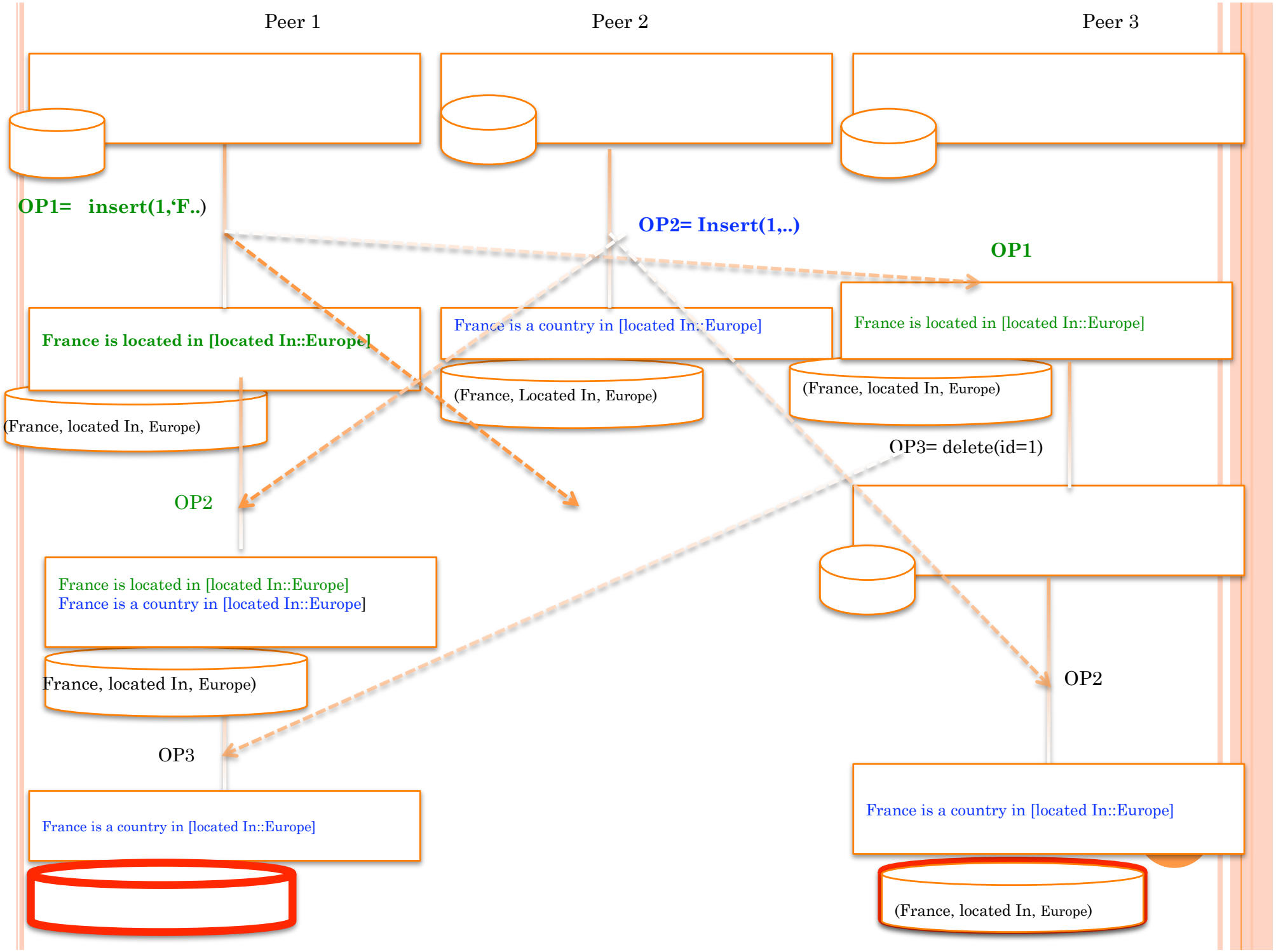
Nejdl et al. Edutella: A P2P networking infrastucture based on RDF. 11th international conference on World Wide Web (WWW), 2002.

# Summary

- P2P Semantic Web replication algorithm
  - Sharing immutable RDF meta-data
  - Concurrent updates not handled
  - Do not ensure CCI Model

# CCI Model for SWOOKI

- Causality same as WOOKI
  - Pre-conditions on editing operation
  - SWOOKI does not introduce new editing operations
- Convergence for text (by WOOT) and semantic data (??)
- Intention preservation:
  - Intention: text (insert ($lp < l < ln$)), semantic data (?)
  - Preservation: text (WOOT), semantic data(?)


Peer 1
Peer 2
Peer 3
OP1= insert(1,'F..)
OP2= Insert(1,..)
OP1
France is located in [located In::Europe]
France is a country in [located In::Europe]
France is located in [located In::Europe]
(France, located In, Europe)
(France, Located In, Europe)
(France, located In, Europe)
OP3= delete(id=1)
OP2
France is located in [located In::Europe]
France is a country in [located In::Europe]
France, located In, Europe)
OP2
OP3
France is a country in [located In::Europe]
France is a country in [located In::Europe]
(France, located In, Europe)

# Intention Violation

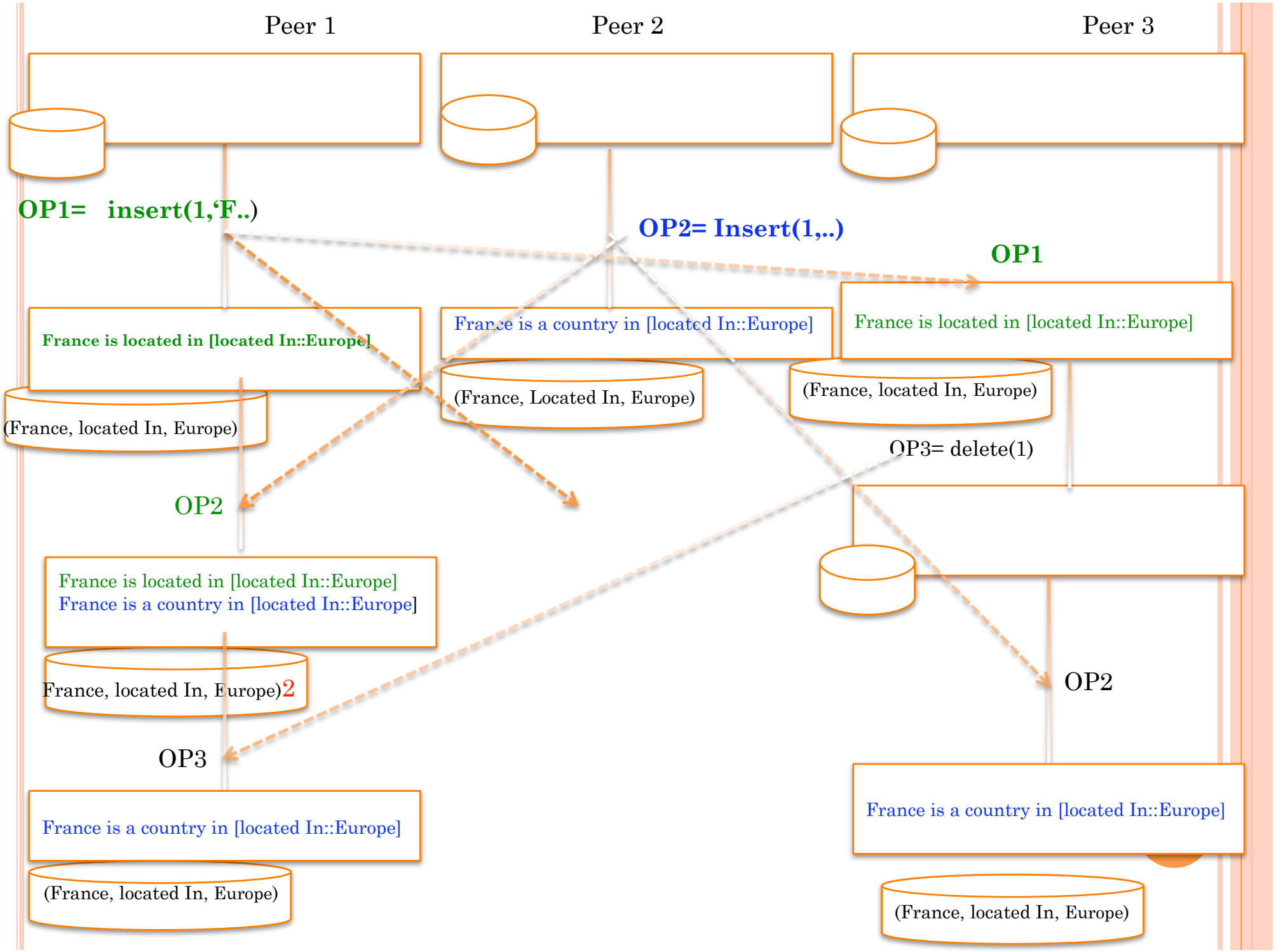
- The effects of the insertion of a triple in a RDF repository is not visible because of concurrent delete
  - The effects of every delete must be visible in RDF repository
  - The effects of every insert must be visible in RDF repository
- Transform RDF repository into multi-set repository

# Insert Intention

$$
\begin{aligned}
\exists i \quad & \wedge \\
& \wedge \exists\, i_P < i \; LineID(Page'[i_P]) = p \\
& \wedge \exists\, i_N \leq i \; LineID(Page'[i_N]) = n \\
& \wedge Page'[i] = newline \\
& \wedge \forall j < i \; Page'[j] = Page[j] \\
& \wedge \forall j \geq i \; Page'[j] = Page[j-1] \\
& \wedge R' \leftarrow R \uplus T
\end{aligned}
$$

# Delete Intention

$$\exists i \ \wedge PageID(Page'[i]) = l$$
$$\wedge \ visibility(Page'[i]) \leftarrow false$$
$$\wedge \ R' \leftarrow R - T$$


Peer 1
Peer 2
Peer 3
OP1= insert(1,'F..)
OP2= Insert(1,..)
OP1
France is located in [located In::Europe]
France is a country in [located In::Europe]
France is located in [located In::Europe]
(France, located In, Europe)
(France, Located In, Europe)
(France, located In, Europe)
OP3= delete(1)
OP2
France is located in [located In::Europe]
France is a country in [located In::Europe]
France, located In, Europe)2
OP2
OP3
France is a country in [located In::Europe]
(France, located In, Europe)
France is a country in [located In::Europe]
(France, located In, Europe)

# Algorithms

```
1  Upon Save(page, oldPage) :-
2      let P ← Diff(page, oldPage)
3   for each op ∈ P do
4      Receive(op)
5   endfor
6  Broadcast(P)
```

```
7  Upon Receive(op) :-
8   if isExecutable(op) then
9      if type(op) = insert then
10       IntegrateIns(op)
11     if type(op) = delete then
12       IntegrateDel(op)
13  else
14       waitingLog ← waitingLog ⋃ {op}
15  endif
```

# Integration

```
37  IntegrateIns(PageID, line, l_P , l_N) :-
38      IntegratedInsT(PageID, line, l_P , l_N)
39      IntegrateInsRDF(line)
```

# Text Integration

```
41  IntegrateInsT(PageID, line, l_P , l_N) :−
42  let S'  ← subseq(Page[PageID]), l_P, l_N)
43  if S    = ∅ then
44    insert(PageID, line, l_N)
45  else
46          let i   ← 0
47          let   d_min ← min(degree(S'))
48          let F ← filter(S', degree = d_min)
49          while (i < |F| − 1) and (F[i] <_id l) do
50              i   ← i +1
51           IntegrateInsT(PageID, line, F[i−1], F[i])
```

# Integration the insertion of RDF data

```
52 IntegrateInsRDF(line) :−
53 let   S ← ExtractRDF(line)
54 if  S ≠ ∅ then
55   for each  triple  ∈ S do
56      if   Contains(triple)  then
57           triple .counter++
58      else
59          insertRDF(R,triple)
60      endif
61  endif
```

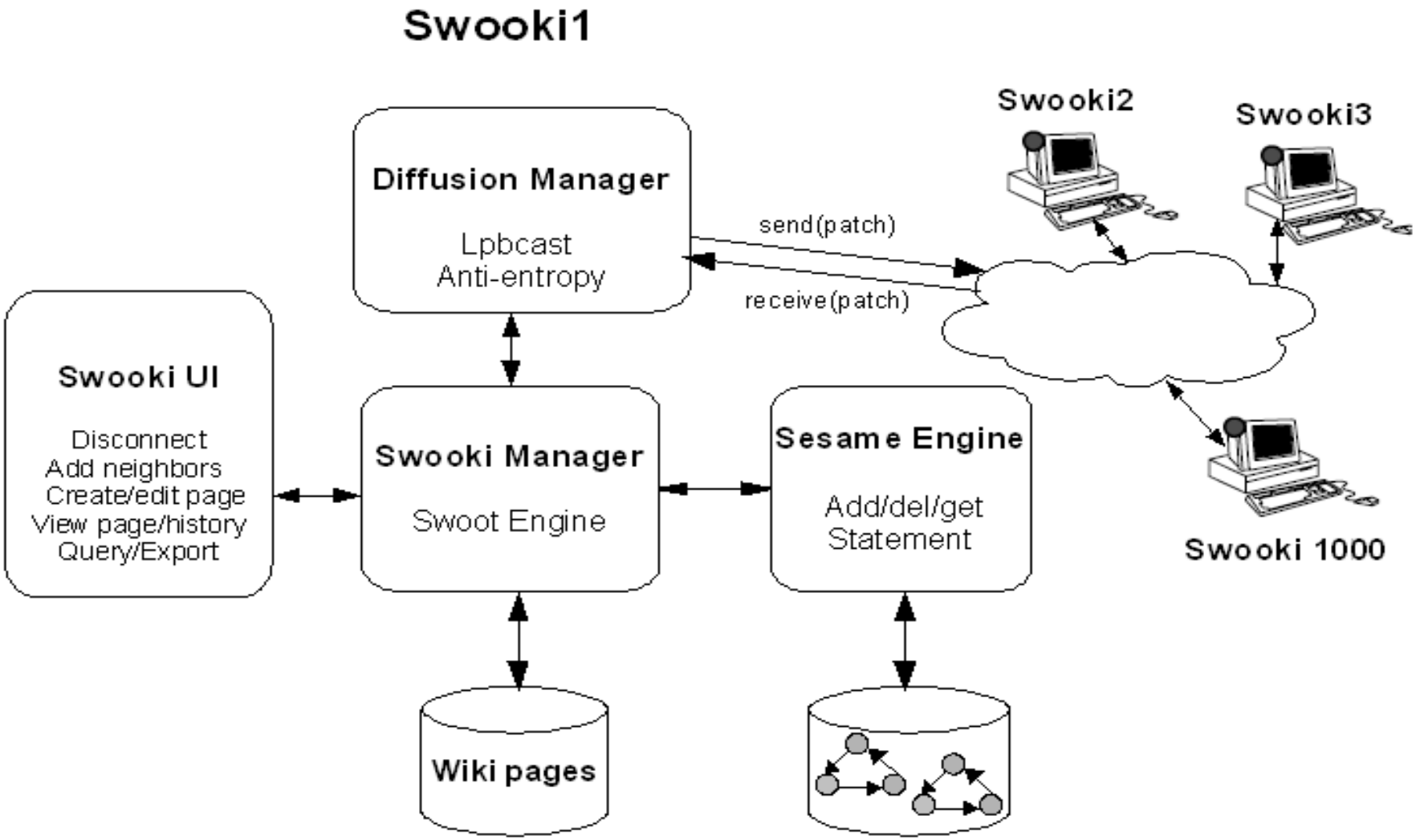
# Integration of delete operation

```
23 IntegrateDel(LineID) :−
24     IntegrateDelT(LineID)
25     IntegrateDelRDF(LineID)
```

```
26 IntegrateDelT(LineID) :−
27     Page[LineID]. visibility    ← false
```

```
28 IntegrateDelRDF(LineID) :−
29 let   S ← ExtractRDF(LineID)
30 if  S ≠ ∅ then
31  for each  triple  ∈ S do
32        triple .counter−−
33      if    triple .counter == 0 then
34        deleteRDF(R,triple)
35      endif
36 endif
```

# SWOOKI ARCHITECTURE




格Woo-Ki氣
http://wooki.loria.fr/wooki2
Menu
Home
Create Page
Site status
Disconnect Wooki
Log
Neighbors table
2 neighbors
Pages List
Home
France
Italy
SPARQL Query
SPARQL query
RDF
RDF Search
Skin
Blue
Ubuntu
Red
France
[View] [Edit] [Source] [History] [Log] [RDFGraph]
France is located in Europe
The capital of France is Paris
Facts
locatedIn : Europe
hasCapital : Paris
Export RDF
Wooki © 2006 - ECOO Team - LORIA
Semantic Query

# Query

```
Select ?s ?p ?v where { ?s ?p ?v.
Filter(regex(str(?s),\
request.getParameter("search")+"\") ||
regex(str(?p),\""+request.getParameter("search")
+"\")" ||
regex(str(?v), \""+request.getParameter("search")
+"\"))}
```

# Conclusion

- SWOOKI is the first P2P Semantic Wikis
- SMW + WOOKI
- CCI Model for SWOOKI data type
- New collaborative modes
  - Off-line work
  - Transactional changes
  - Collborative knowledge building

# Conclusion

- Availability, fault-tolerance, load-balancing
  - Locality transparency
- Performance
  - Execution query: local execution
  - Messages delivery: LpbCast , one round, WOOT does need extra messages
- Data synchronization:
  - Convergence in one round, no extra messages for integration
  - Complexity : $O(n2)$

# Open Issues and Perspectives

- Security
  - In wikis security policies presented as attribute of the page
  - Replicate security policies
  - Use the same approach to hand this new data type (operations, CCI Model)
- Distributed queries to reduce the load and share cost
- SMW + partial replication (reduce traffics, infinite storage)
- IkeWiki + WOOKI
- IkeWiki + partial replication