Extending the VoDKA Architecture with P2P Aggregated Content Management

Carlos Abalde\textsuperscript{1} Víctor M. Gulías\textsuperscript{1} Laura M. Castro\textsuperscript{1}
\{cabalde, gulias, lcastro\}@udc.es

\textsuperscript{1}MADS Group - Department of Computer Science
University of A Coruña - Spain

IEEE Symposium on Computers and Communications
July 1-4, 2007 • Aveiro (Portugal)
Outline

1. Problem overview
   - What is VoDKA?
   - What do we want to do with VoDKA?
   - What is the problem?
   - Content distribution networks (CDNs)

2. Screens coordination
   - State dissemination & replication
   - Coordination algorithm

3. Resource scheduling
   - Resource scheduling agent
   - Resource scheduling algorithm

4. Prototype benchmarking

5. Conclusions & Future work
Outline

1. Problem overview
   - What is VoDKA?
   - What do we want to do with VoDKA?
   - What is the problem?
   - Content distribution networks (CDNs)

2. Screens coordination
   - State dissemination & replication
   - Coordination algorithm

3. Resource scheduling
   - Resource scheduling agent
   - Resource scheduling algorithm

4. Prototype benchmarking

5. Conclusions & Future work
What is VoDKA?

- VoDKA: Video on Demand Kernel Architecture
- Project started on 1999 and is almost entirely based on a declarative language (Erlang)
- Suitable for IPTV deployments, TVoDSL or DVB-C networks, internet streaming, and content delivery providers
- Multiprotocol, multiprovider, modularity, flexibility, scalability...

VoDKA: Video on Demand Kenel Architecture

- Project started on 1999 and is almost entirely based on a declarative language (Erlang)
- Suitable for IPTV deployments, TVoDSL or DVB-C networks, internet streaming, and content delivery providers
- Multiprotocol, multiprovider, modularity, flexibility, scalability...

What do we want to do with VoDKA?

Network of public LAN-connected video-information screens

- Airport, rail station, mall, university...
- Media objects (MOs) stored in a VoDKA system
  - xDSL link $\Rightarrow$ MOs locally cached in video-information screens
- Media schedule / timetable $\Rightarrow$ Deadline timestamps

Requirements

- Minimize screens coupling
- Centralized screen/location priority framework
- Avoid single points of failure
- Avoid scalability limitations
- Allow efficient screen departure and arrivals
- Minimize deployment and maintenance costs
What do we want to do with VoDKA?

Network of public LAN-connected video-information screens

- Airport, rail station, mall, university...
- Media objects (MOs) stored in a VoDKA system
  - xDSL link ⇒ MOs locally cached in video-information screens
- Media schedule / timetable ⇒ Deadline timestamps

Requirements

\[
\begin{align*}
\text{Minimize screens coupling} \\
\text{Centralized screen/location priority framework} \\
\text{Avoid single points of failure} \\
\text{Avoid scalability limitations} \\
\text{Allow efficient screen departure and arrivals} \\
\text{Minimize deployment and maintenance costs}
\end{align*}
\]
What is VoDKA?
What do we want to do with VoDKA?
What is the problem?

Content distribution networks (CDNs)

Direct connection approach

- xDSL link overload ⇒
  More expensive connection?
- VoDKA overload ⇒
  Larger deployment?

+ Simplicity
- Unscalability
- High deployment cost
Direct connection approach

- xDSL link overload $\Rightarrow$ More expensive connection?
- VoDKA overload $\Rightarrow$ Larger deployment?

- Simplicity
- Unscalability
- High deployment cost
Direct connection approach

- xDSL link overload ⇒ More expensive connection?
- VoDKA overload ⇒ Larger deployment?

+ Simplicity
- Unscalability
- High deployment cost
Direct connection approach

- **xDSSL link overload** ⇒ More expensive connection?
- **VoDKA overload** ⇒ Larger deployment?

+ Simplicity
- Unscalability
- High deployment cost
Proxy approach

Highly-correlated server access patterns → Request aggregation

- Proxy nodes coordinating LAN-access to VoDKA
- Local VoDKA caching agent
- Single point of failure
- High maintenance cost
Proxy approach

Highly-correlated server access patterns ⇒ Request aggregation

- Proxy nodes coordinating LAN-access to VoDKA
  + Local VoDKA caching agent
  - Single point of failure
  - High maintenance cost
Proxy approach

Highly-correlated server access patterns $\Rightarrow$ Request aggregation

- Proxy nodes coordinating LAN-access to VoDKA
- Local VoDKA caching agent
- Single point of failure
- High maintenance cost
$N$ screens/LAN, all with identical media scheduling ($S$ GB/screen) ⇒

- **Direct connection**
  - $N \times S$ VoDKA-LAN GB transferred

- **Proxy**
  - $S$ VoDKA-LAN GB transferred
  - Independent of $N$

Per-location distributed scheduling

- Decentralized screen coordination algorithm required
- Broadcast? ⇒ Unscalability
- Multicast? ⇒ Maintenance cost ~ network configuration

P2P coordination algorithm based on a DHT
So...?

$N$ screens/LAN, all with identical media scheduling ($S$ GB/screen) \(\Rightarrow\)

- Direct connection
  - $N \times S$ VoDKA-LAN GB transferred

- Proxy
  + $S$ VoDKA-LAN GB transferred
  + Independent of $N$

- Per-location distributed scheduling
  - Decentralized screen coordination algorithm required
  - Broadcast? \(\Rightarrow\) Unscalability
  - Multicast? \(\Rightarrow\) Maintenance cost $\sim$ network configuration

P2P coordination algorithm based on a DHT
$N$ screens/LAN, all with identical media scheduling ($S$ GB/screen) $\Rightarrow$

- Direct connection
  - $N \times S$ VoDKA-LAN GB transferred

- Proxy
  - $S$ VoDKA-LAN GB transferred
  - Independent of $N$

Per-location distributed scheduling

- Decentralized screen coordination algorithm required
- Broadcast? $\Rightarrow$ Unscalability
- Multicast? $\Rightarrow$ Maintenance cost $\sim$ network configuration

P2P coordination algorithm based on a DHT
Content distribution networks (CDNs)

- Publishing a huge quantity of relevant information to be accessed by a large number of clients.
- Powerful, flexible, scalable and fault tolerant architectures are demanded.
- Two general approaches are identified:
  - Infrastructure-based content distribution: Improvement of the server infrastructure in a traditional client/server framework (e.g., Akamai).
  - P2P content distribution: Replacement of the client/server model by a P2P approach (e.g., Napster or Gnutella). More effort in terms of coordination, resource management, heterogeneity...
Content distribution networks (CDNs)

- Publishing a huge quantity of relevant information to be accessed by a large number of clients
- Powerful, flexible, scalable and fault tolerant architectures are demanded
- Two general approaches are identified

Infrastructure-based content distribution

Improvement of the server infrastructure in a traditional client/server framework (e.g., Akamai)

P2P content distribution

Replacement of the client/server model by a P2P approach (e.g., Napster or Gnutella). More effort in terms of coordination, resource management, heterogeneity...
Outline

1. Problem overview
   - What is VoDKA?
   - What do we want to do with VoDKA?
   - What is the problem?
   - Content distribution networks (CDNs)

2. Screens coordination
   - State dissemination & replication
   - Coordination algorithm

3. Resource scheduling
   - Resource scheduling agent
   - Resource scheduling algorithm

4. Prototype benchmarking

5. Conclusions & Future work
One DHT for each LAN is defined. MO identifiers are used as keys, storing information about the state of MO copies in the local network.

- Global state dissemination & replication
- MO pointer + MO state
- Coordination algorithm based on global state
State dissemination & replication

One DHT for each LAN is defined. MO identifiers are used as keys, storing information about the state of MO copies in the local network.

- Global state dissemination & replication
  - MO pointer + MO state
- Coordination algorithm based on global state
One DHT for each LAN is defined. MO identifiers are used as keys, storing information about the state of MO copies in the local network.

- Global state dissemination & replication
  - MO pointer + MO state
- Coordination algorithm based on global state

State dissemination & replication

Carlos Abalde, Víctor M. Gulías, Laura M. Castro
Extending VoDKA with P2P Aggregated Content Management - ISCC’07
Node architecture

Outline
- Problem overview
- Screens coordination
- Resource scheduling
- Prototype benchmarking
- Conclusions & Future work

State dissemination & replication

Coordination algorithm

Carlos Abalde, Víctor M. Gulías, Laura M. Castro

Extending VoDKA with P2P Aggregated Content Management - ISCC’07
Coordination algorithm (I)

if $\text{stg}(MO) \in \{\text{DOWNLOADING}, \text{TRANSFERRING}\}$ then
  $\text{vodka.notify}(MO, \text{tst})$
else if $\text{stg}(MO) \equiv \text{PENDING}$ then
  if $\text{copies} = \text{dstate}(MO)$ then
    $\text{vodka.notify}(MO, \text{tst})$
    $\text{node} = \text{select_peer}(\text{copies})$
    $\text{node.transfer}(MO)$
    $\text{stg}(MO) = \text{dstate}(MO) = \{\text{TRANSFERRING, node}\}$
  else
    $\text{vodka.download}(MO, \text{tst})$
    $\text{stg}(MO) = \text{dstate}(MO) = \text{DOWNLOADING}$
  end if
end if
Coordination algorithm (II)

+ No screens coupling
  - Local (neighbor) screens unreachable (worst case) ⇒ DHT becomes HT ⇒ Direct connection approach
+ No single points of failure
  - Distributed, balanced and replicated MOs metainformation
+ No scalability limits
  - Based on DHT design
    - Ready to handle huge number of nodes
    - Low latency lookup
+ Efficient screen departure and arrivals
  - Based on DHT design (DHTs ensure minimal disturbing)
  - Lightweight MOs metainformation (pointers + state) simplifies DHT data transfers
+ Minimal deployment and maintenance costs
  - Fully decentralized P2P architecture
  - Same LAN link & VoDKA deployment
Coordination algorithm (and III)

State dissemination & replication

Coordinating algorithm

Carlos Abalde, Víctor M. Gulías, Laura M. Castro

Extending VoDKA with P2P Aggregated Content Management - ISCC’07
Outline

1. Problem overview
   - What is VoDKA?
   - What do we want to do with VoDKA?
   - What is the problem?
   - Content distribution networks (CDNs)

2. Screens coordination
   - State dissemination & replication
   - Coordination algorithm

3. Resource scheduling
   - Resource scheduling agent
   - Resource scheduling algorithm

4. Prototype benchmarking

5. Conclusions & Future work
VoDKA as an agent-based system

- Agent-based system
  - Media storage, control, protocol adaptation, caching, replication, supervision...
- New resource scheduling agent
Resource scheduling agent

- Resource scheduling algorithm \(\Rightarrow\) share VoDKA resources among a set of LANs \(L_i\)
  - Dynamic programming problem
  - Maximize LAN-link usage
  - Minimize out-of-time MOs

- Based on,
  - Screen/location priority framework \((p(L_i))\)
  - LAN-link bandwidth \((bw_{\text{max}}(L_i))\)
  - Coordination algorithm notifications (MO deadlines)

- Periodical bitrate tuning in order to fulfill screen scheduling deadlines

- VoDKA - screen transferences multiplexed in a single connection \(\Rightarrow\) minimize server connections
Resource scheduling algorithm (I)

When is it executed?

- $p(L_i)$ or $bw_{max}(L_i)$ is tuned by a VoDKA administrator
- A new download request is received
- A timestamp update is received

Based on,

- $bw_{min}(L_i) = \sum_{\text{deadline} - \text{now}(i)} \frac{\text{size} - \text{pos}}{\text{deadline} - \text{now}(i)} \equiv$ LAN $L_i$ demanded bandwidth
- $bw_{max}(L_i) \equiv$ LAN $L_i$ priority
- $p(L_i) \equiv$ LAN $L_i$ link bandwidth

In charge of calculating $bw(L_i)$ for all LAN $L_i$

- If $bw(L_i) < bw_{min}(L_i) \Rightarrow$ out-of-time MOs
Resource scheduling algorithm (and II)

1. Build transferences table

<table>
<thead>
<tr>
<th>$L_i$</th>
<th>$L_{i'}$</th>
<th>MO</th>
<th>size</th>
<th>pos</th>
<th>deadline</th>
<th>BW</th>
</tr>
</thead>
<tbody>
<tr>
<td>$L_1$</td>
<td>$L_1$</td>
<td>MO/32</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>?</td>
</tr>
<tr>
<td></td>
<td>$L_1$</td>
<td>MO/7</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>?</td>
</tr>
<tr>
<td></td>
<td>$L_2$</td>
<td>MO/12</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>?</td>
</tr>
<tr>
<td>$L_2$</td>
<td>$L_1$</td>
<td>MO/12</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>?</td>
</tr>
<tr>
<td>$L_3$</td>
<td>$L_3$</td>
<td>MO/4</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>?</td>
</tr>
<tr>
<td></td>
<td>$L_3$</td>
<td>MO/17</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>?</td>
</tr>
</tbody>
</table>

2. Compress table

<table>
<thead>
<tr>
<th>LAN</th>
<th>$bw_{\min}(L_i)$</th>
<th>$bw_{\max}(L_i)$</th>
<th>$p(L_i)$</th>
<th>$bw(L_i)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$L_1$</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>?</td>
</tr>
<tr>
<td>$L_2$</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>?</td>
</tr>
<tr>
<td>$L_3$</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>?</td>
</tr>
</tbody>
</table>

3. Run scheduling algorithm

- $\forall i, bw(L_i) = 0$
- $bw_{\text{remaining}} = bw_{\text{VoDKA}}$

\[ \text{if } \sum bw_{\min}(L_i) \leq bw_{\text{VoDKA}} \text{ then} \]

\[ bw(L_i) = \min(bw_{\min}(L_i), bw_{\max}(L_i)) \]

\[ bw_{\text{remaining}} = bw_{\text{remaining}} - bw(L_i) \]

\[ \text{end if} \]

\[ \text{free} = \{i \mid bw(L_i) < bw_{\max}(L_i)\} \]

while $\text{free} \neq [] \land bw_{\text{remaining}} > 0$ do

for all $i \in \text{free}$ do

\[ np = p(L_i) / \sum p(L_i) / i \in \text{free} \]

\[ bw(L_i) = bw(L_i) + bw_{\text{remaining}} \times np \]

\[ bw_{\text{remaining}} = bw_{\text{remaining}} - bw_{\text{remaining}} \times np \]

if $bw(L_i) > bw_{\max}(L_i)$ then

\[ \text{excess} = bw(L_i) - bw_{\max}(L_i) \]

\[ bw(L_i) = bw_{\max}(L_i) \]

\[ bw_{\text{remaining}} = bw_{\text{remaining}} + \text{excess} \]

Remove $i$ from free

end if

end for

end while

4. Share $bw(L_i)$ optimal values among $L_i$ transferences

5. Update bitrates
Outline

1. Problem overview
   - What is VoDKA?
   - What do we want to do with VoDKA?
   - What is the problem?
   - Content distribution networks (CDNs)

2. Screens coordination
   - State dissemination & replication
   - Coordination algorithm

3. Resource scheduling
   - Resource scheduling agent
   - Resource scheduling algorithm

4. Prototype benchmarking

5. Conclusions & Future work
Prototype benchmarking

- Erlang + Chord
- Same scheduling
- Synchronized scheduling update

- 128 screens ~ 1.2 screens
- Pathological behavior

Programme size

LAN screens

LAN-screen equivalence

Carlos Abalde, Víctor M. Gulías, Laura M. Castro

Extending VoDKA with P2P Aggregated Content Management - ISCC'07
Outline

1. Problem overview
   - What is VoDKA?
   - What do we want to do with VoDKA?
   - What is the problem?
   - Content distribution networks (CDNs)

2. Screens coordination
   - State dissemination & replication
   - Coordination algorithm

3. Resource scheduling
   - Resource scheduling agent
   - Resource scheduling algorithm

4. Prototype benchmarking

5. Conclusions & Future work
Conclusions & Future work

- Real world development based on a direct application of DHTs
  - Interesting approach for highly-correlated large-scale CDNs
  - Integration in a real world streaming server
- DHTs are a powerful building block in decentralized behavior design

Future work

- Distributed P2P caching architecture based on DHTs
- Resource management & lookup in DHTs
Extending the VoDKA Architecture with P2P Aggregated Content Management

Carlos Abalde¹  Víctor M. Gulías¹  Laura M. Castro¹
{cabalde, gulias, lcastro}@udc.es

¹MADS Group - Department of Computer Science
University of A Coruña - Spain

IEEE Symposium on Computers and Communications
July 1-4, 2007 • Aveiro (Portugal)
A glimpse of Erlang

- Designed in Ericsson’s Computer Science Lab (CSLab)
  - Ericsson ⇒ AXD301 ATM switch
  - Nortel Networks (Alteon) ⇒ SSL accelerator
  - MADS Group ⇒ VoDKA project
  - …
- Distributed functional paradigm (simple and easy to learn)
  - High level of abstraction
  - Built in solid concurrency model
  - Explicit or transparent distribution
  - Asynchronous message passing
- Soft real time
  - Lightweight processes, fast message passing and efficient garbage collection
- Robustness
  - Simple and consistent error recovery model
  - Supervision hierarchies
  - Online code upgrading support
P2P content distribution

Fully decentralized and structured P2P architectures are a better scalable design which make them more suitable for a CDN design.

- Two common problems are identified in these architectures:
  - Overlay network creation and maintenance
    - How to build peers structure without any centralized servers or supernodes?
    - How to keep this mechanism scalable and fault tolerant?
  - Content lookup
    - How to find an item in a P2P system in a scalable way, without any centralized servers?
- Both questions are addressed using distributed hash tables (DHTs)
Fully decentralized and structured P2P architectures are a better scalable design which make them more suitable for a CDN design.

- Two common problems are identified in these architectures:
  - Overlay network creation and maintenance
    - How to build peers structure without any centralized servers or supernodes?
    - How to keep this mechanism scalable and fault tolerant?
  - Content lookup
    - How to find an item in a P2P system in a scalable way, without any centralized servers?

- Both questions are addressed using distributed hash tables (DHTs)
P2P content distribution

Fully decentralized and structured P2P architectures are a better scalable design which make them more suitable for a CDN design.

- Two common problems are identified in these architectures:
  1. **Overlay network creation and maintenance**
     - How to build peers structure without any centralized servers or supernodes?
     - How to keep this mechanism scalable and fault tolerant?
  2. **Content lookup**
     - How to find an item in a P2P system in a scalable way, without any centralized servers?

- Both questions are addressed using *distributed hash tables (DHTs)*.
P2P content distribution

- P2P networks are formed on top of the underlying physical computer network, usually referred to as an overlay network.

Classification,

- Centralization degree:
  - Purely decentralized architectures
  - Partially centralized architectures
  - Hybrid decentralized architectures

- Internal structure:
  - Unstructured
  - Structured

<table>
<thead>
<tr>
<th>Centralization</th>
<th>Hybrid</th>
<th>Partial</th>
<th>None</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unstructured</td>
<td>Napster</td>
<td>Kazaa</td>
<td>Gnutella</td>
</tr>
<tr>
<td>Structured</td>
<td></td>
<td></td>
<td>Chord, CAN, Tapestry, Pastry</td>
</tr>
</tbody>
</table>
P2P Content Distribution: Centralization Degree

- **On purely decentralized architectures**, all nodes in the network perform the same tasks, acting both as servers and clients, and there is no central coordination of activities.

- **Partially centralized architectures** have the same principles but they include special nodes –supernodes– which assume a more important role, acting as local central indexes for files shared by local peers.

- **Hybrid decentralized architectures** have a central server to ease interaction between peers; although end-to-end interaction and file exchanges may take place directly between peers.
P2P Content Distribution: Internal Structure

- **Unstructured** systems locate the content unrelatedly to the overlay topology. Searching mechanisms are based on network flooding with propagating queries
  - Joe Armstrong wrote, “*Gnutella is like wandering into a random room in a random town in a random country and shouting at a random group of passers by “anybody got a fish”*”
  - More appropriate for accommodating highly transient node populations
  - Scalability limitations

- **Structured** systems are build over a tightly controlled overlay topology. They provide a mapping between content and location in the form of a distributed routing table
  - Scalable solution for exact-match (id) queries
  - Main disadvantage: overlay network building and maintenance
Distributed hash tables (DHTs)

A DHT is a hash table distributed in a set of nodes providing an interface to store and retrieve key/value pairs

- Different overlay networks
- Contents -files- are stored in the DHT
- CAN, Chord, Tapestry, Pastry, Kademlia, Koorde…
- DHT algorithms stress the ability to,
  - Scale well to huge numbers of nodes
  - Locate keys with low latency (usually, $O(\log N)$)
  - Handle node arrivals and departures scalably
  - Ease the maintenance of per-node routing tables
  - Balance the distribution of keys
Distributed hash tables: Chord

Succ. Table

<table>
<thead>
<tr>
<th>i</th>
<th>id+2</th>
<th>succ</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>6</td>
</tr>
</tbody>
</table>

Items 7

query(7)

Succ. Table

<table>
<thead>
<tr>
<th>i</th>
<th>id+2</th>
<th>succ</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>6</td>
</tr>
</tbody>
</table>

Items 1

Succ. Table

<table>
<thead>
<tr>
<th>i</th>
<th>id+2</th>
<th>succ</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>6</td>
</tr>
</tbody>
</table>

Items 6

Succ. Table

<table>
<thead>
<tr>
<th>i</th>
<th>id+2</th>
<th>succ</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>