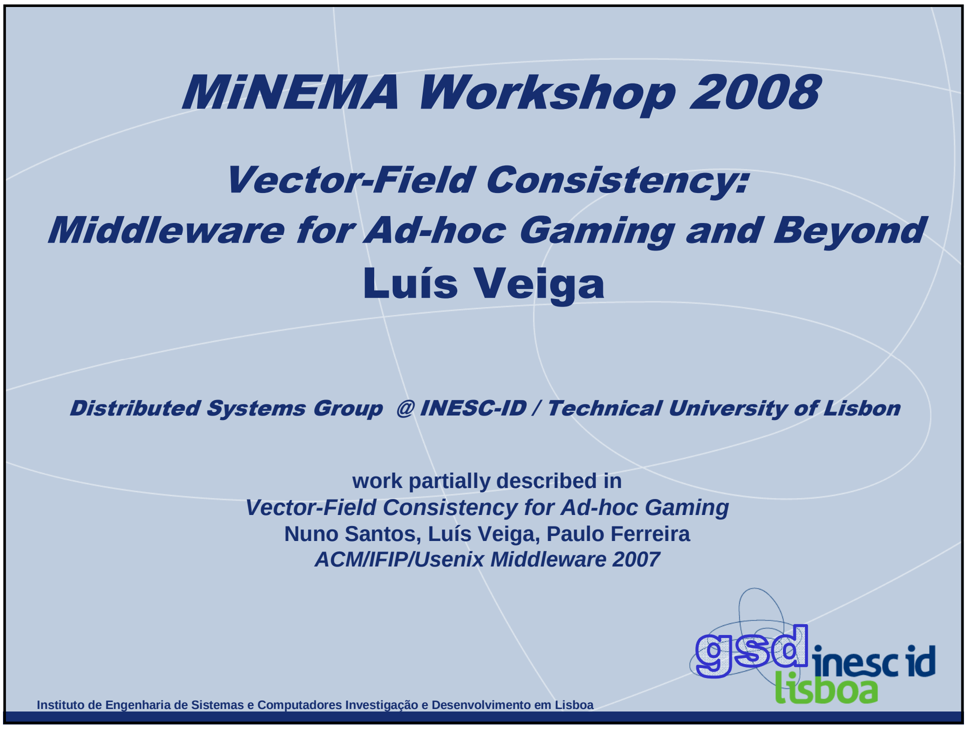


MiNEMA Workshop 2008

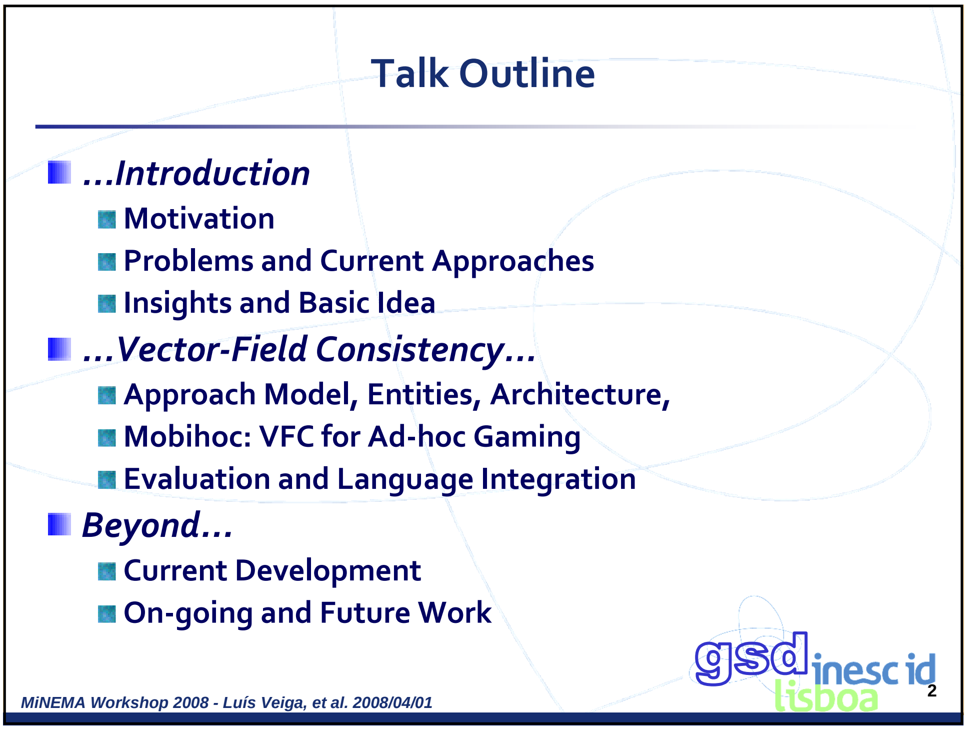
Vector-Field Consistency:
Middleware for Ad-hoc Gaming and Beyond

Luís Veiga

Distributed Systems Group @ INESC-ID / Technical University of Lisbon

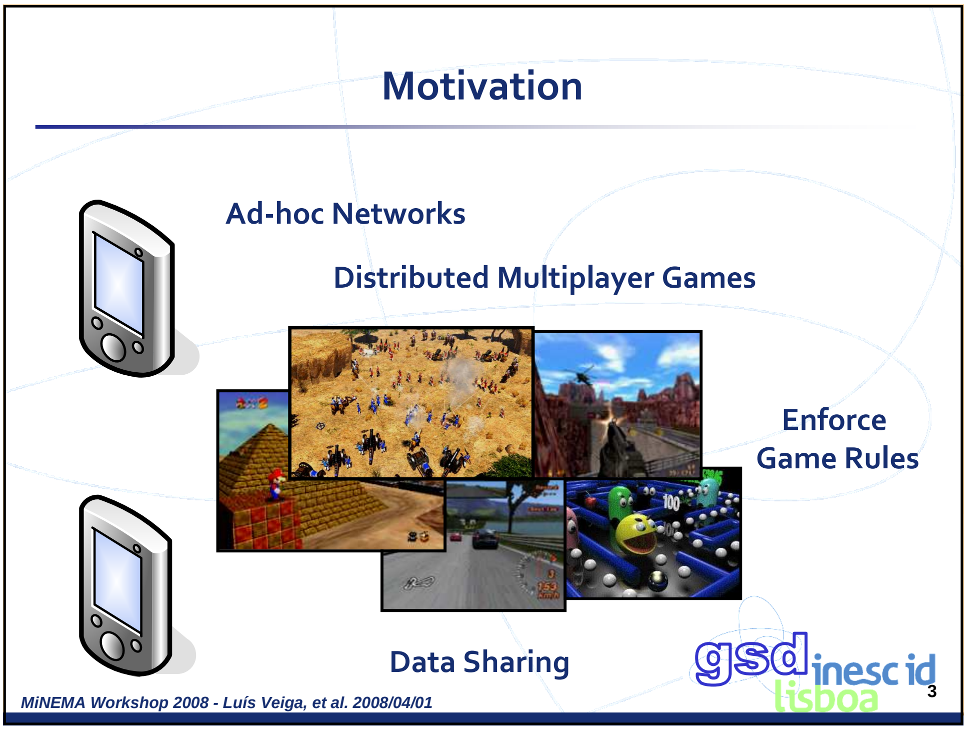
work partially described in
Vector-Field Consistency for Ad-hoc Gaming
Nuno Santos, Luís Veiga, Paulo Ferreira
ACM/IFIP/Usenix Middleware 2007

Instituto de Engenharia de Sistemas e Computadores Investigação e Desenvolvimento em Lisboa



Talk Outline

- *...Introduction*
  - Motivation
  - Problems and Current Approaches
  - Insights and Basic Idea
- *...Vector-Field Consistency...*
  - Approach Model, Entities, Architecture,
  - Mobihoc: VFC for Ad-hoc Gaming
  - Evaluation and Language Integration
- *Beyond...*
  - Current Development
  - On-going and Future Work



Motivation

Ad-hoc Networks

Distributed Multiplayer Games

Enforce Game Rules

Data Sharing

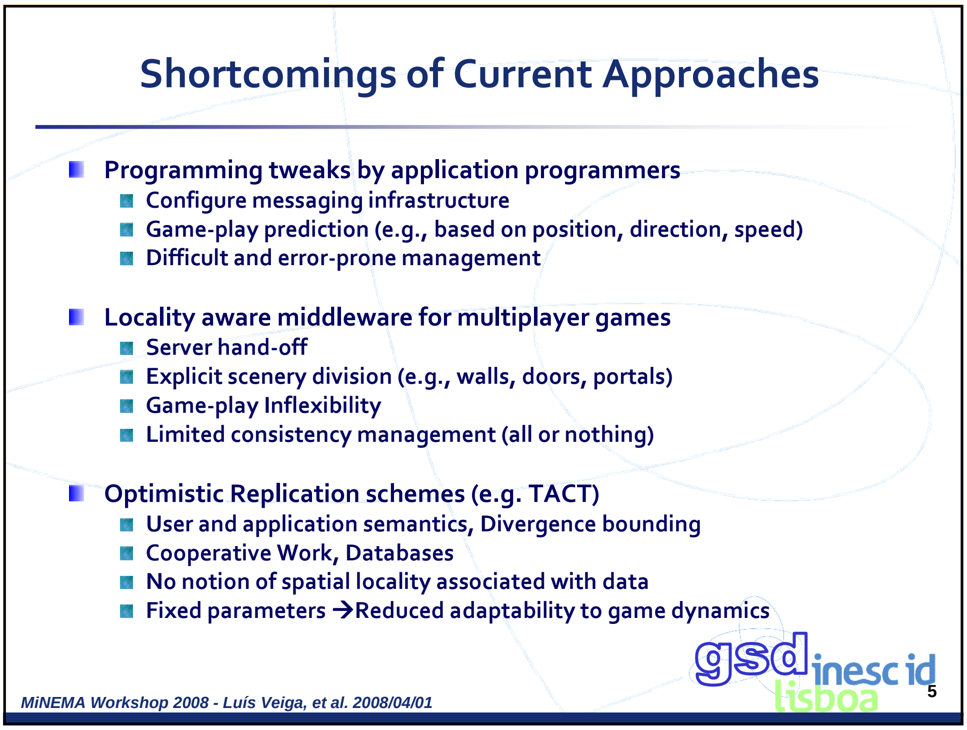
Problem: Consistency vs Playability Tradeoff

P1
P3
P2
P1
P3
P4
P2
P4

Strong Consistency (Update Propagation)

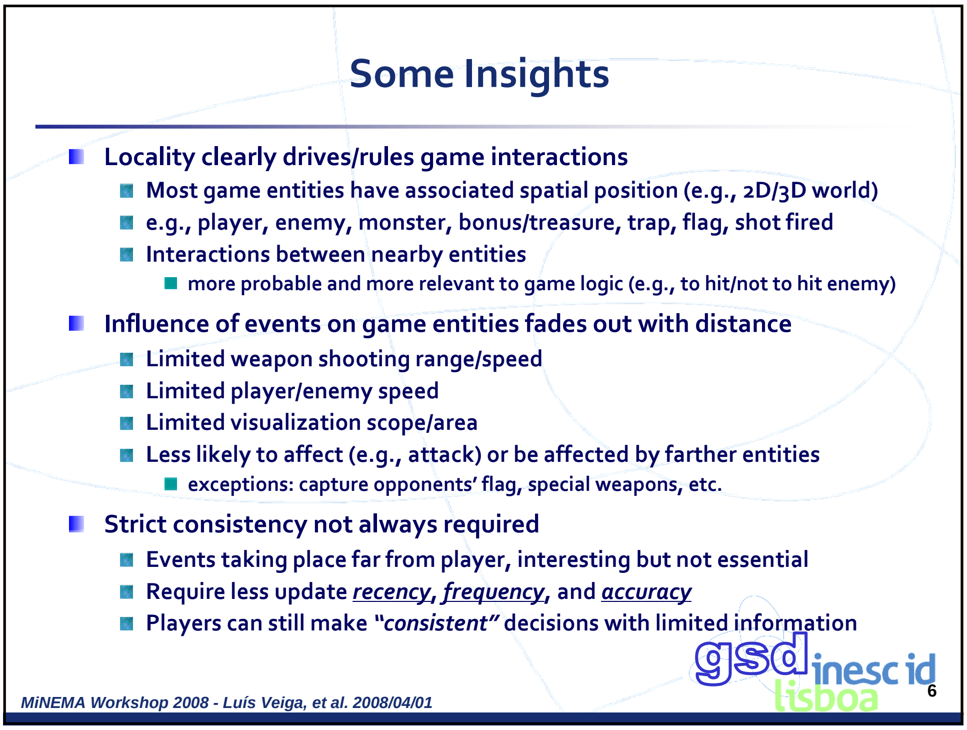
Hard!

Good Playability (Speed/Fluidity, Scope)



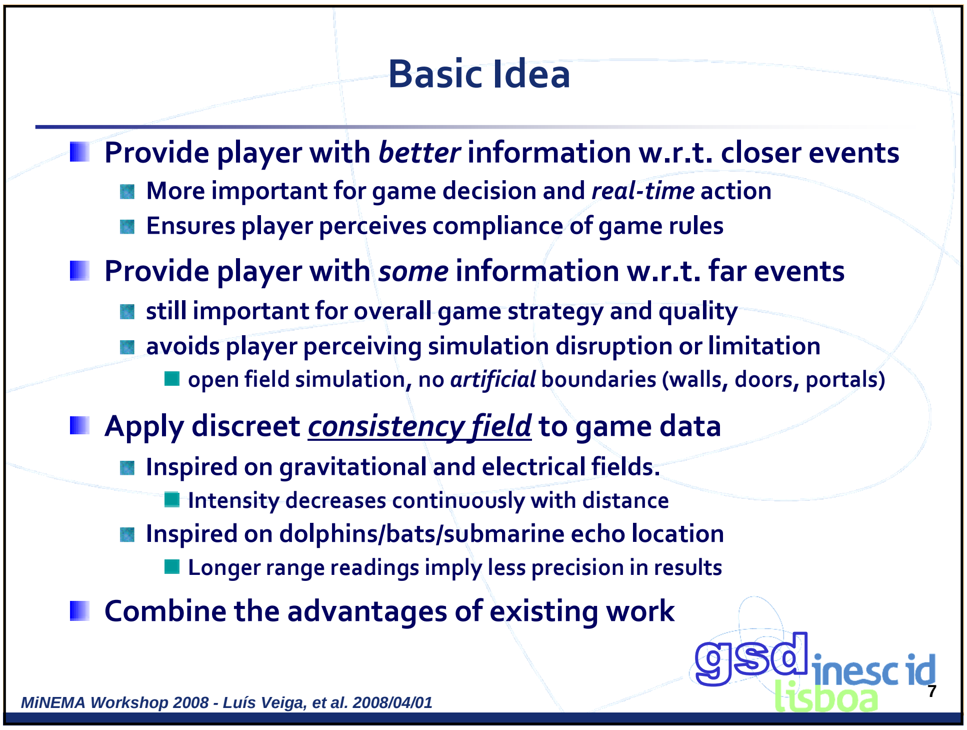
Shortcomings of Current Approaches

- Programming tweaks by application programmers
  - Configure messaging infrastructure
  - Game-play prediction (e.g., based on position, direction, speed)
  - Difficult and error-prone management
- Locality aware middleware for multiplayer games
  - Server hand-off
  - Explicit scenery division (e.g., walls, doors, portals)
  - Game-play Inflexibility
  - Limited consistency management (all or nothing)
- Optimistic Replication schemes (e.g. TACT)
  - User and application semantics, Divergence bounding
  - Cooperative Work, Databases
  - No notion of spatial locality associated with data
  - Fixed parameters →Reduced adaptability to game dynamics



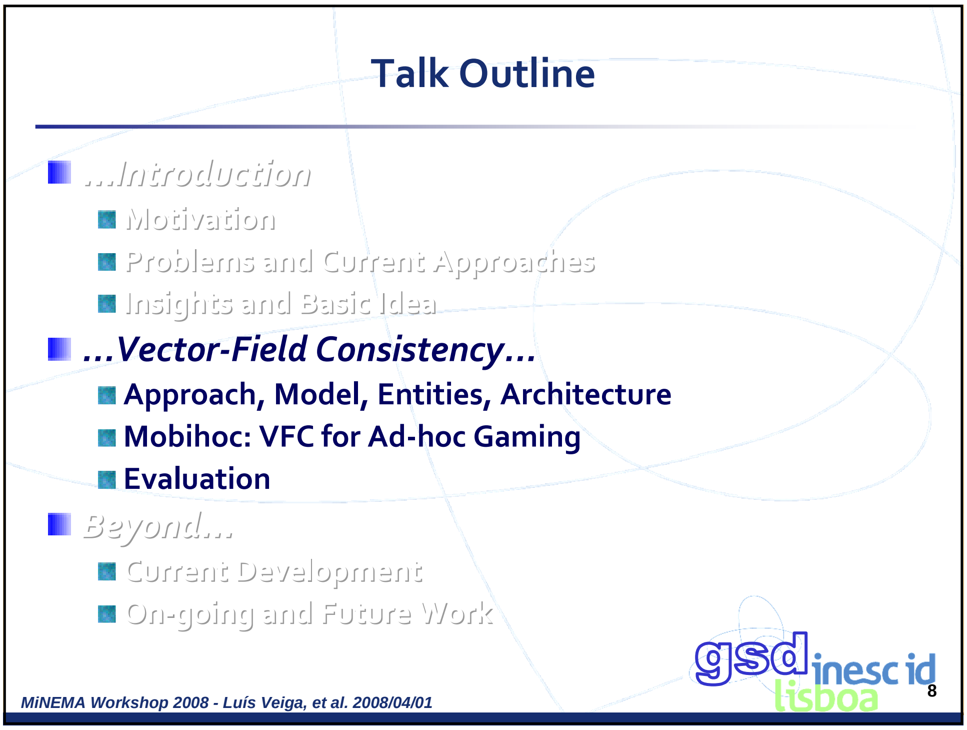
Some Insights

- Locality clearly drives/rules game interactions
  - Most game entities have associated spatial position (e.g., 2D/3D world)
  - e.g., player, enemy, monster, bonus/treasure, trap, flag, shot fired
  - Interactions between nearby entities
    - more probable and more relevant to game logic (e.g., to hit/not to hit enemy)
- Influence of events on game entities fades out with distance
  - Limited weapon shooting range/speed
  - Limited player/enemy speed
  - Limited visualization scope/area
  - Less likely to affect (e.g., attack) or be affected by farther entities
    - exceptions: capture opponents' flag, special weapons, etc.
- Strict consistency not always required
  - Events taking place far from player, interesting but not essential
  - Require less update *recency*, *frequency*, and *accuracy*
  - Players can still make *"consistent"* decisions with limited information



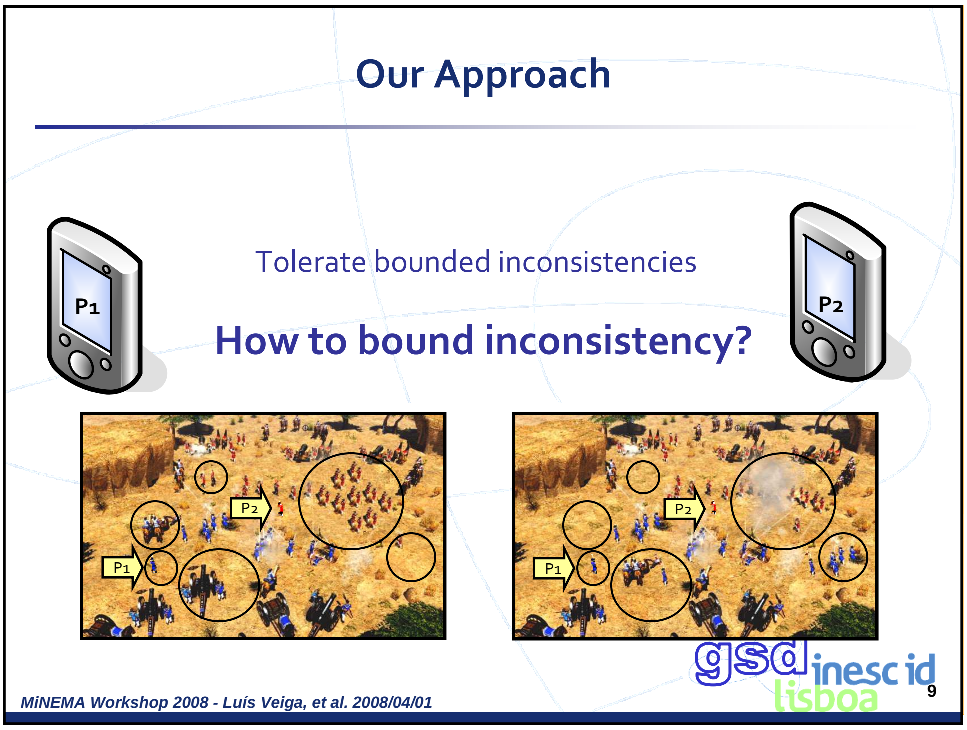
Basic Idea

- Provide player with *better* information w.r.t. closer events
  - More important for game decision and *real-time* action
  - Ensures player perceives compliance of game rules
- Provide player with *some* information w.r.t. far events
  - still important for overall game strategy and quality
  - avoids player perceiving simulation disruption or limitation
    - open field simulation, no *artificial* boundaries (walls, doors, portals)
- Apply discreet *consistency field* to game data
  - Inspired on gravitational and electrical fields.
    - Intensity decreases continuously with distance
  - Inspired on dolphins/bats/submarine echo location
    - Longer range readings imply less precision in results
- Combine the advantages of existing work



Talk Outline

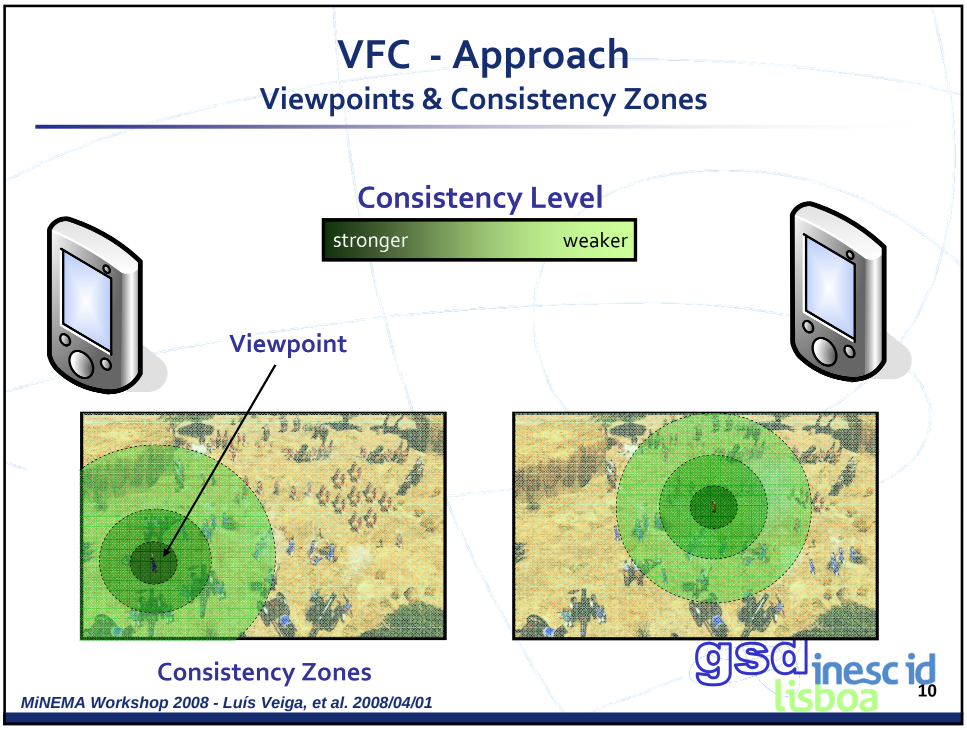
- *…Introduction*
  - Motivation
  - Problems and Current Approaches
  - Insights and Basic Idea
- *…Vector-Field Consistency…*
  - Approach, Model, Entities, Architecture
  - Mobihoc: VFC for Ad-hoc Gaming
  - Evaluation
- *Beyond…*
  - Current Development
  - On-going and Future Work



Our Approach

Tolerate bounded inconsistencies

How to bound inconsistency?



VFC - Approach
Viewpoints & Consistency Zones

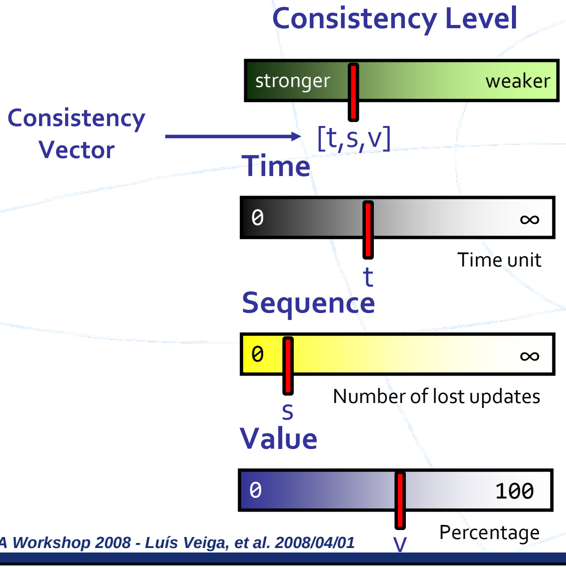
Consistency Level

stronger — weaker

Viewpoint

Consistency Zones

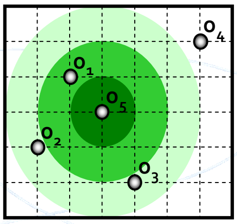
VFC - Consistency Vectors



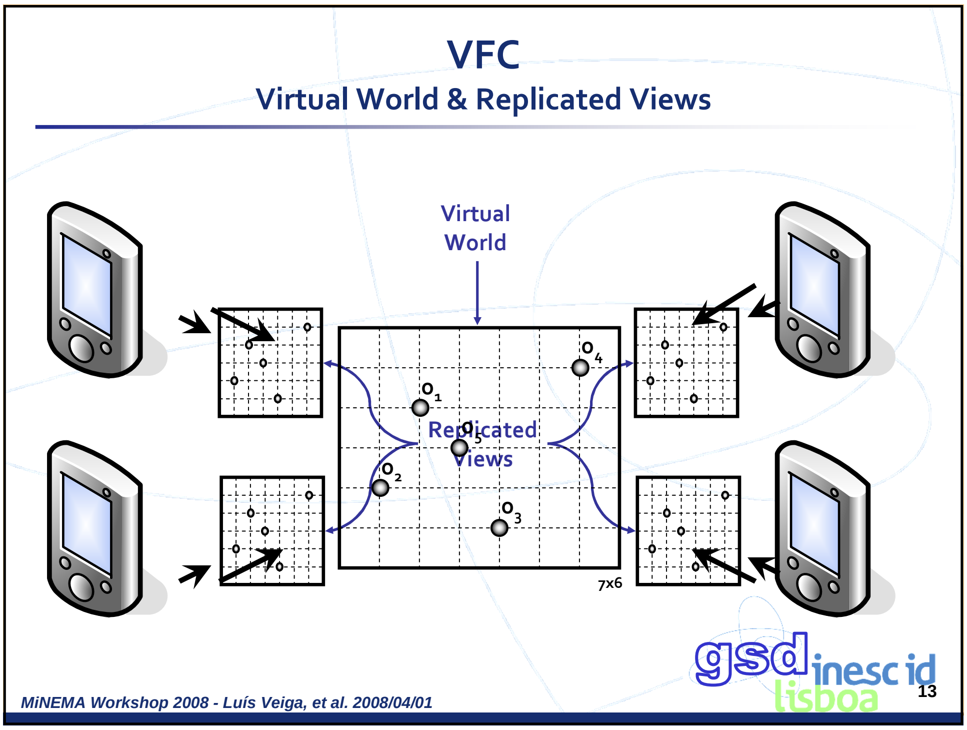
Contribution

Vector-Field Consistency (VFC)

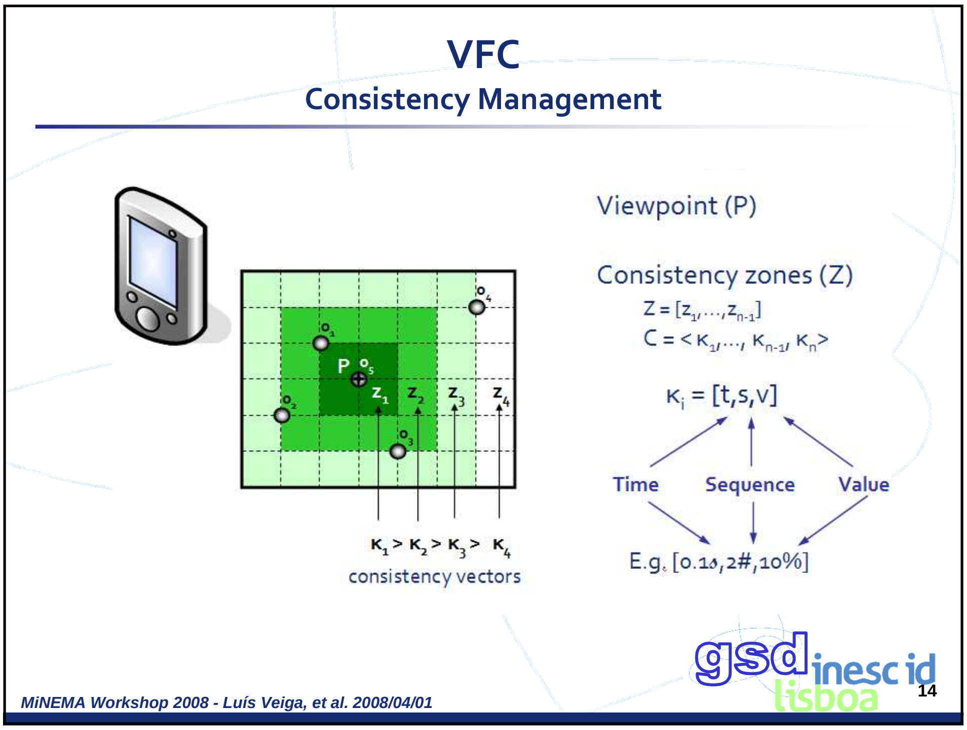
- Consistency model
- Locality-awareness techniques
- Techniques for limiting replica divergence
- Straightforward abstractions
- Mobihoc platform (J2ME)



- Viewpoint
- Consistency zones
- Consistency vectors: [Time, Sequence, Value]



VFC
Virtual World & Replicated Views



VFC
Consistency Management

Viewpoint (P)

Consistency zones (Z)

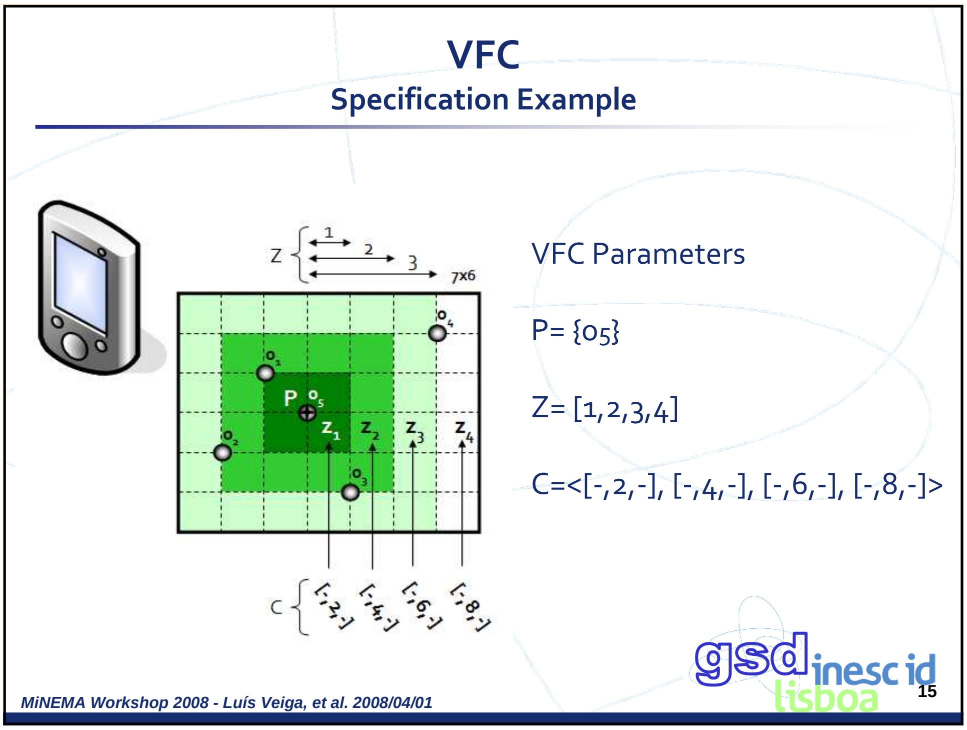
$Z = [z_1, \ldots, z_{n-1}]$

$C = < \kappa_1, \ldots, \kappa_{n-1}, \kappa_n >$

$\kappa_i = [t,s,v]$

Time Sequence Value

E.g. [0.1s,2#,10%]



VFC

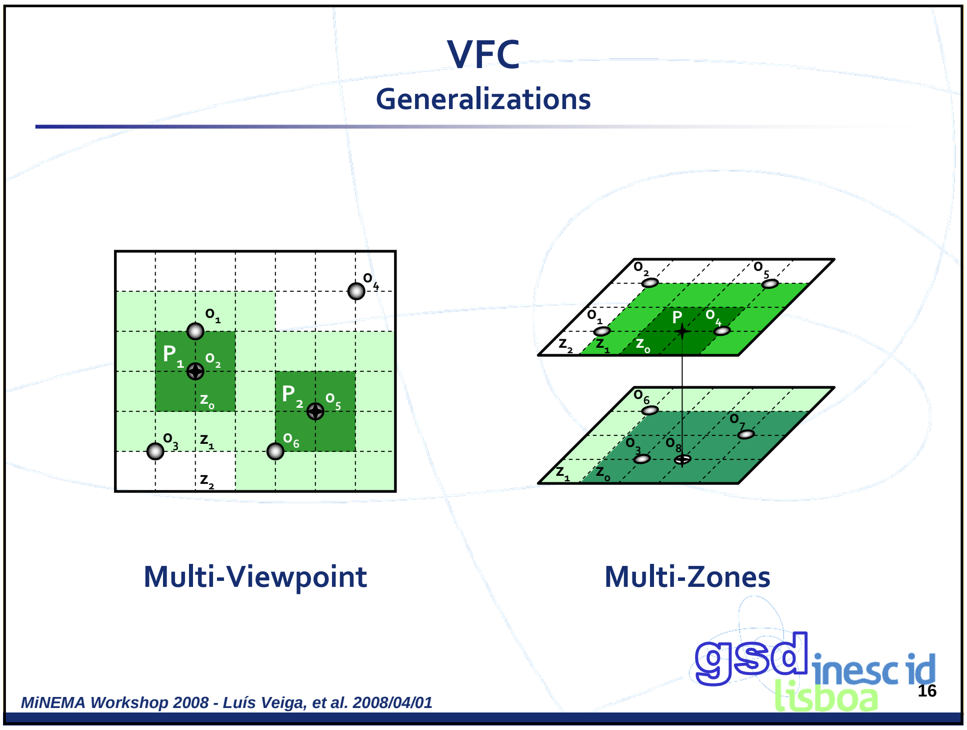
Specification Example

VFC Parameters

P= {o5}

Z= [1,2,3,4]

C=<[-,2,-], [-,4,-], [-,6,-], [-,8,-]>

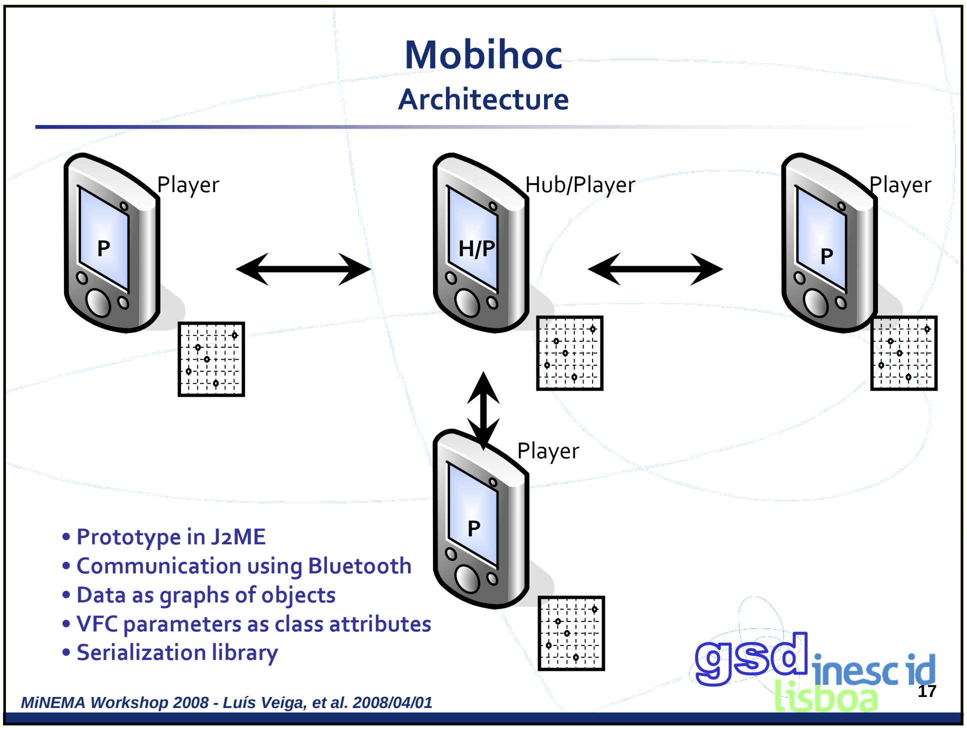


VFC

Generalizations

Multi-Viewpoint

Multi-Zones



Mobihoc

Architecture

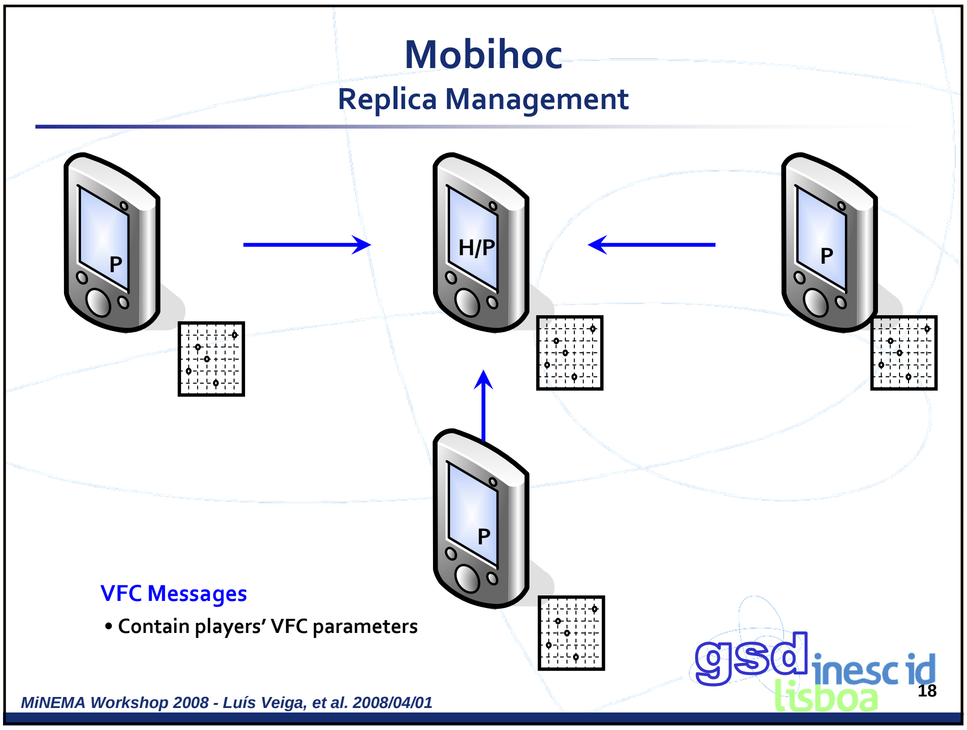
Player

Hub/Player

Player

Player

- Prototype in J2ME
- Communication using Bluetooth
- Data as graphs of objects
- VFC parameters as class attributes
- Serialization library

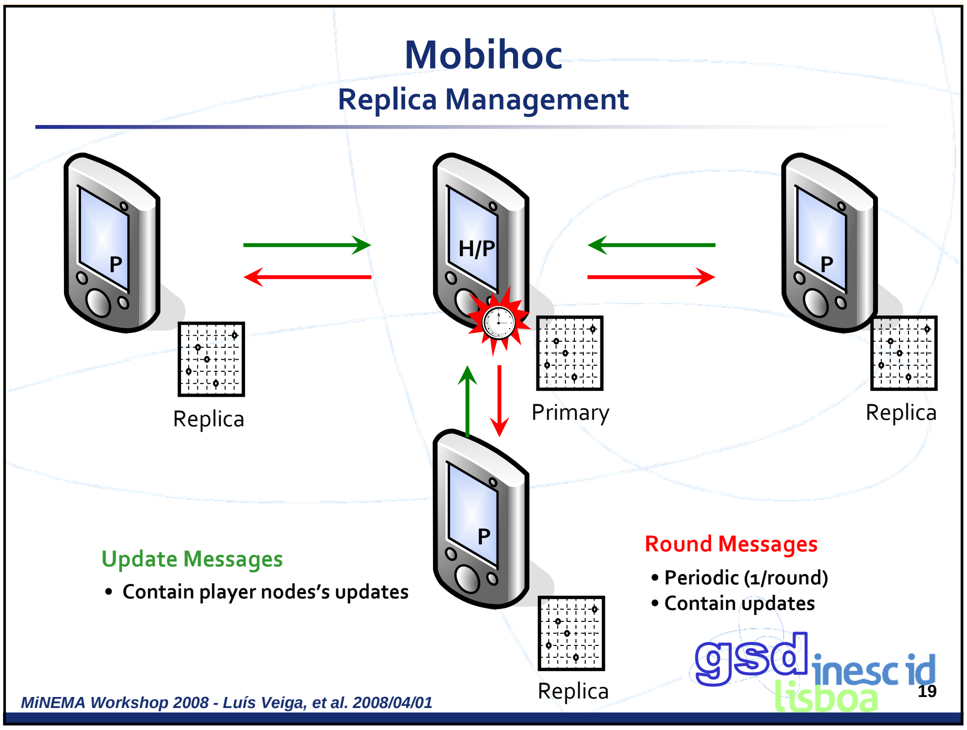


Mobihoc

Replica Management

VFC Messages

- Contain players' VFC parameters



Mobihoc

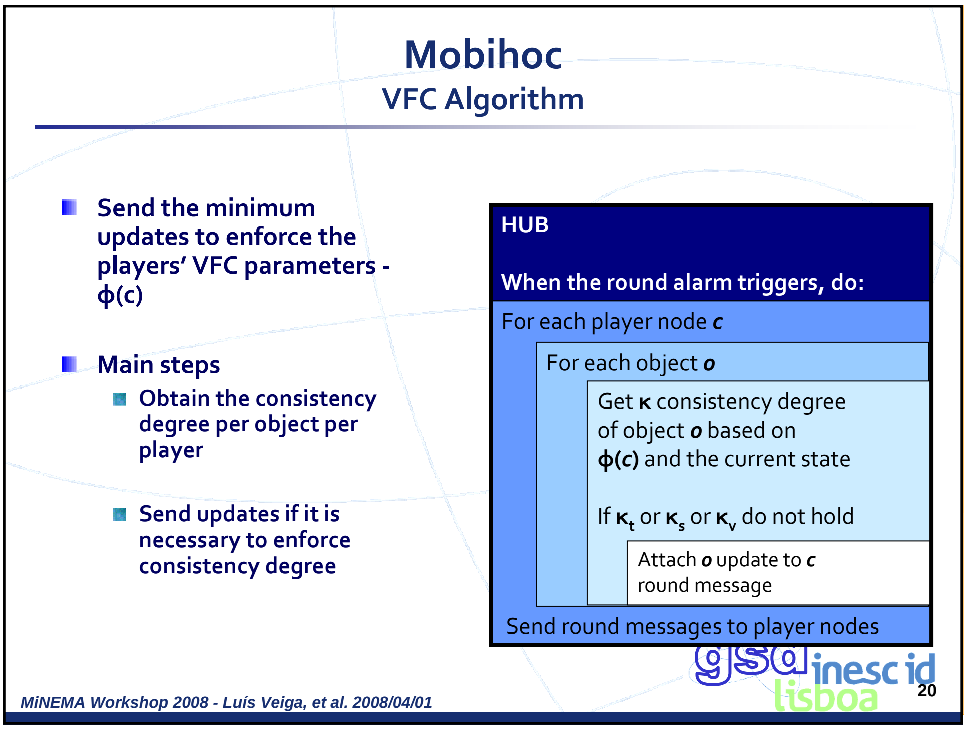
Replica Management

Update Messages

- Contain player nodes's updates

Round Messages

- Periodic (1/round)
- Contain updates



Mobihoc

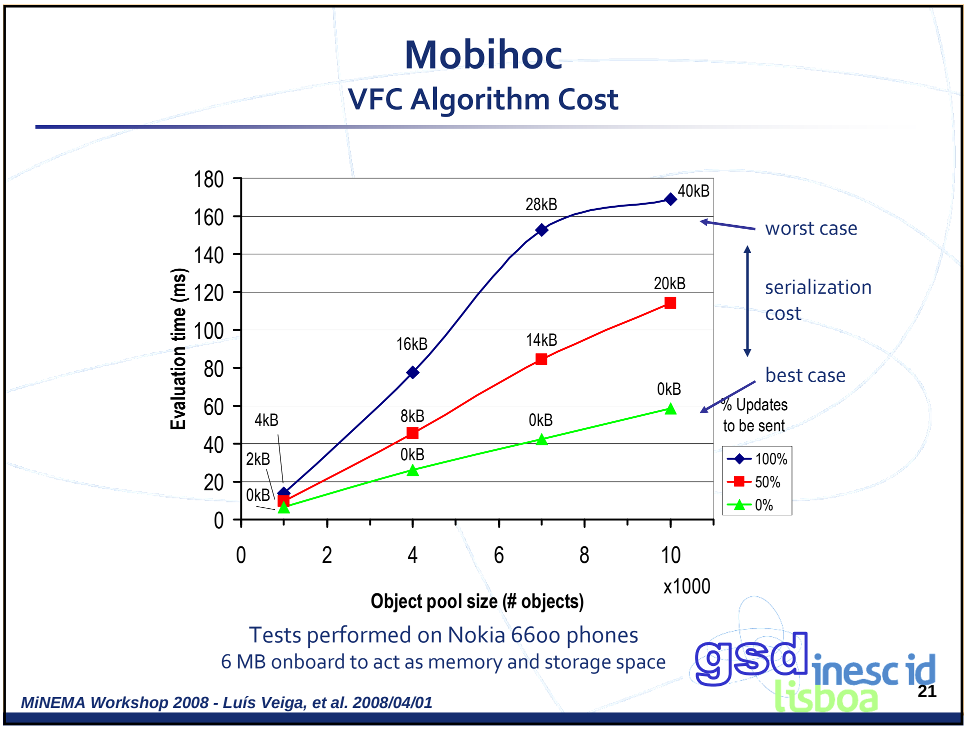
VFC Algorithm

- Send the minimum updates to enforce the players' VFC parameters - ϕ(c)
- Main steps
  - Obtain the consistency degree per object per player
  - Send updates if it is necessary to enforce consistency degree

```
HUB

When the round alarm triggers, do:
For each player node c
    For each object o
        Get κ consistency degree of object o based on ϕ(c) and the current state

        If κt or κs or κv do not hold
            Attach o update to c round message
Send round messages to player nodes
```



Mobihoc

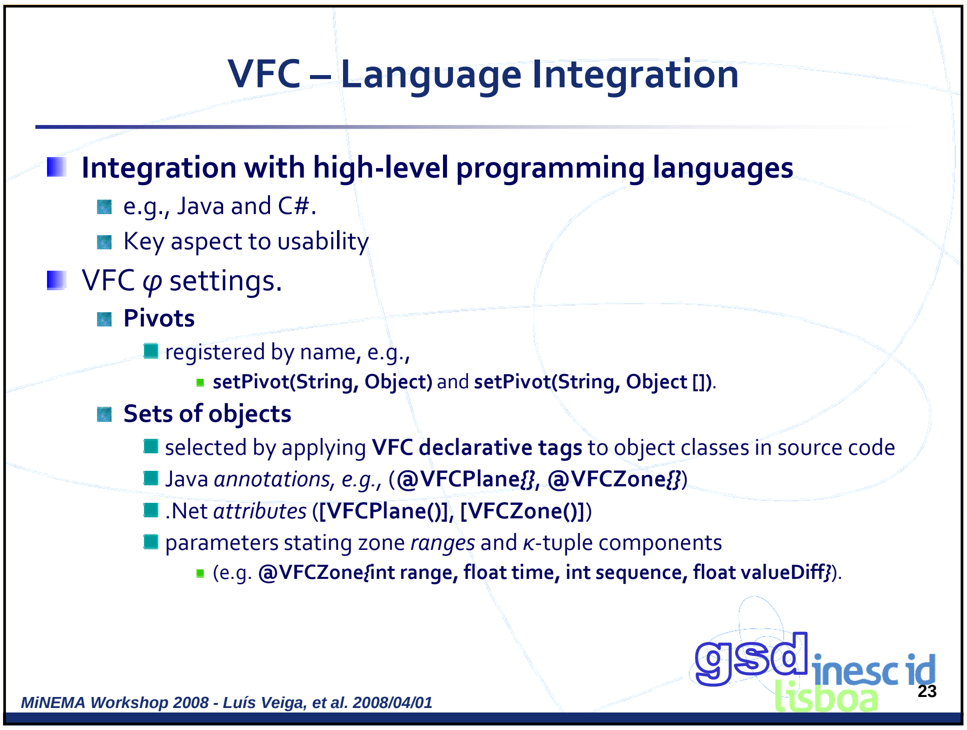
VFC Algorithm Cost

Tests performed on Nokia 66oo phones

6 MB onboard to act as memory and storage space

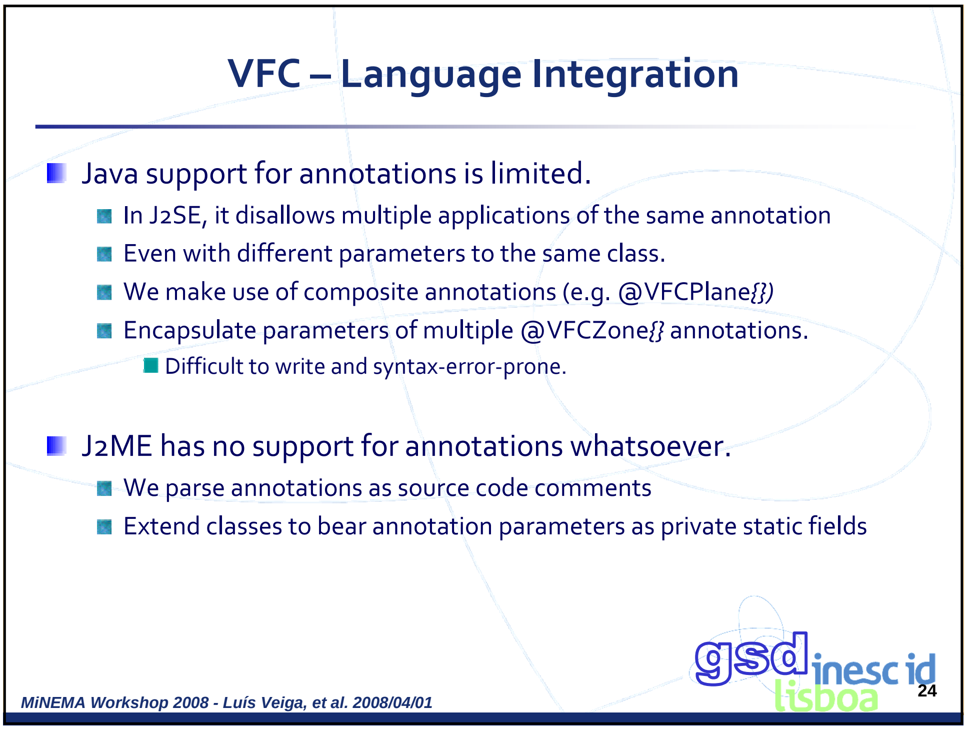
VFC - Multiplayer Pacman Game

- Proof-of-concept
- Distributed multiplayer Pacman
  - Nokia phones
  - Bluetooth
  - 8x8 Maze Grid
- VFC application
  - 3 Zones
    - Room
    - Adjacent Rooms
    - Rest of Grid
  - sequence dimension



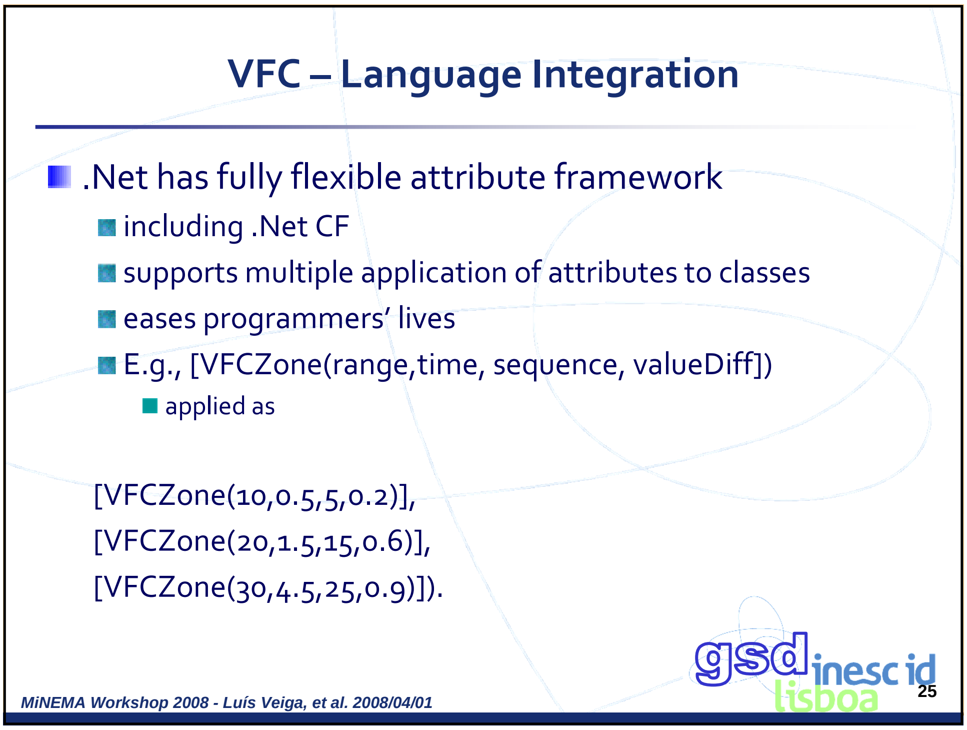
VFC – Language Integration

- **Integration with high-level programming languages**
  - e.g., Java and C#.
  - Key aspect to usability
- VFC φ settings.
  - **Pivots**
    - registered by name, e.g.,
      - **setPivot(String, Object)** and **setPivot(String, Object [])**.
  - **Sets of objects**
    - selected by applying **VFC declarative tags** to object classes in source code
    - Java *annotations, e.g.*, (**@VFCPlane*{}*, @VFCZone*{}***)
    - .Net *attributes* (**[VFCPlane()], [VFCZone()]**)
    - parameters stating zone *ranges* and κ-tuple components
      - (e.g. **@VFCZone*{*int range, float time, int sequence, float valueDiff*}***).



VFC – Language Integration

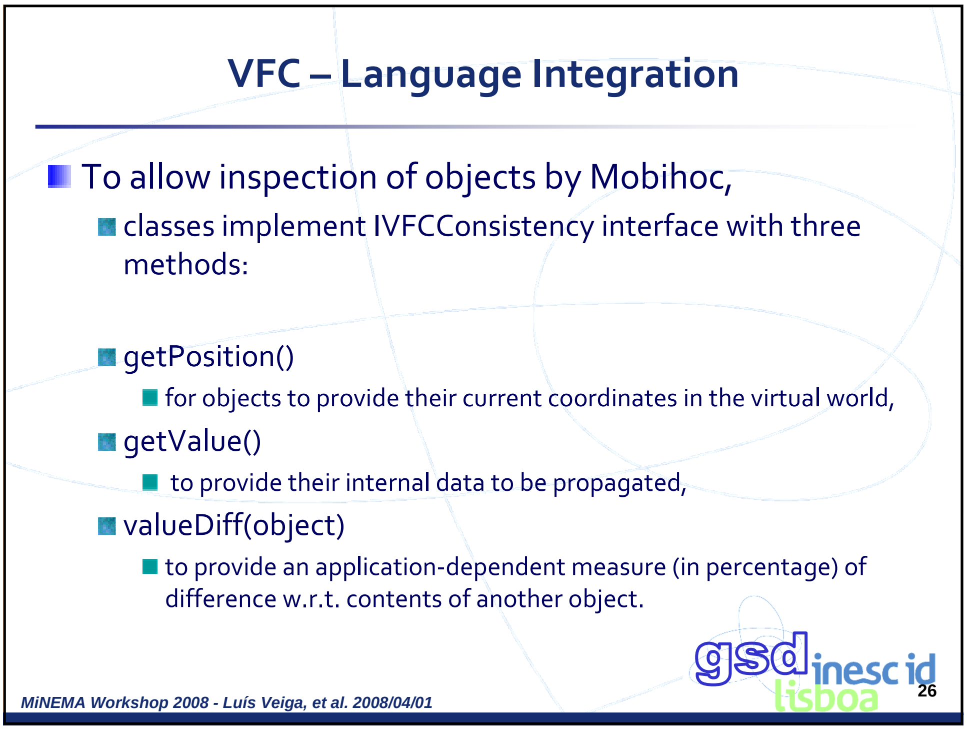
- Java support for annotations is limited.
  - In J2SE, it disallows multiple applications of the same annotation
  - Even with different parameters to the same class.
  - We make use of composite annotations (e.g. @VFCPlane*{})*
  - Encapsulate parameters of multiple @VFCZone*{}* annotations.
    - Difficult to write and syntax-error-prone.
- J2ME has no support for annotations whatsoever.
  - We parse annotations as source code comments
  - Extend classes to bear annotation parameters as private static fields



VFC – Language Integration

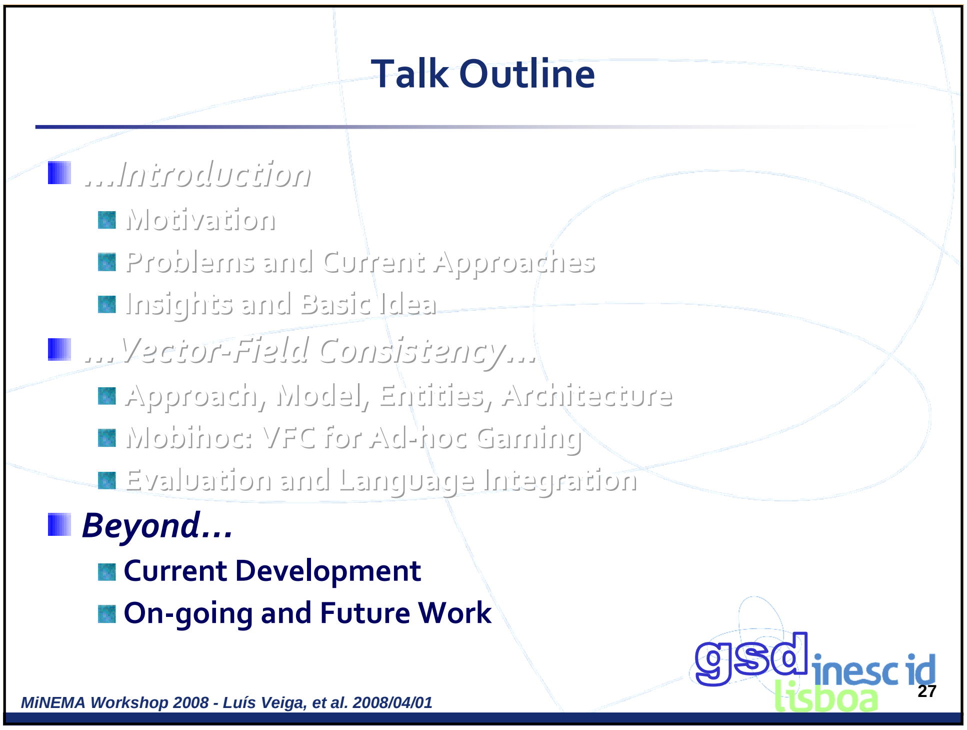
- .Net has fully flexible attribute framework
  - including .Net CF
  - supports multiple application of attributes to classes
  - eases programmers' lives
  - E.g., [VFCZone(range,time, sequence, valueDiff])
    - applied as

[VFCZone(10,0.5,5,0.2)],
[VFCZone(20,1.5,15,0.6)],
[VFCZone(30,4.5,25,0.9)]).



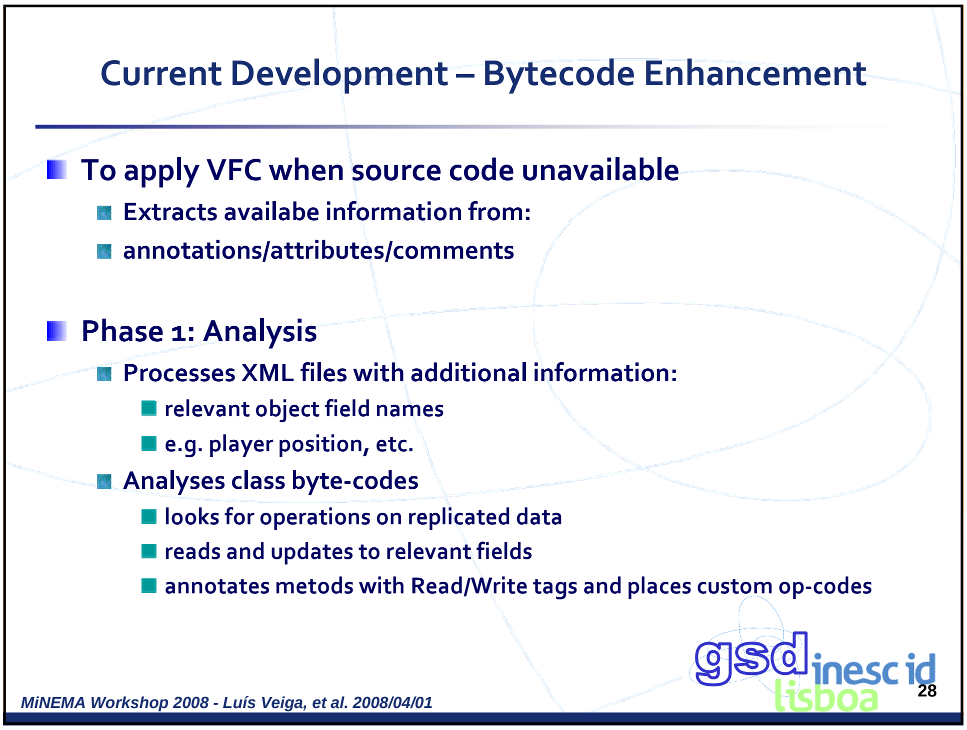
VFC – Language Integration

- To allow inspection of objects by Mobihoc,
  - classes implement IVFCConsistency interface with three methods:
  - getPosition()
    - for objects to provide their current coordinates in the virtual world,
  - getValue()
    - to provide their internal data to be propagated,
  - valueDiff(object)
    - to provide an application-dependent measure (in percentage) of difference w.r.t. contents of another object.



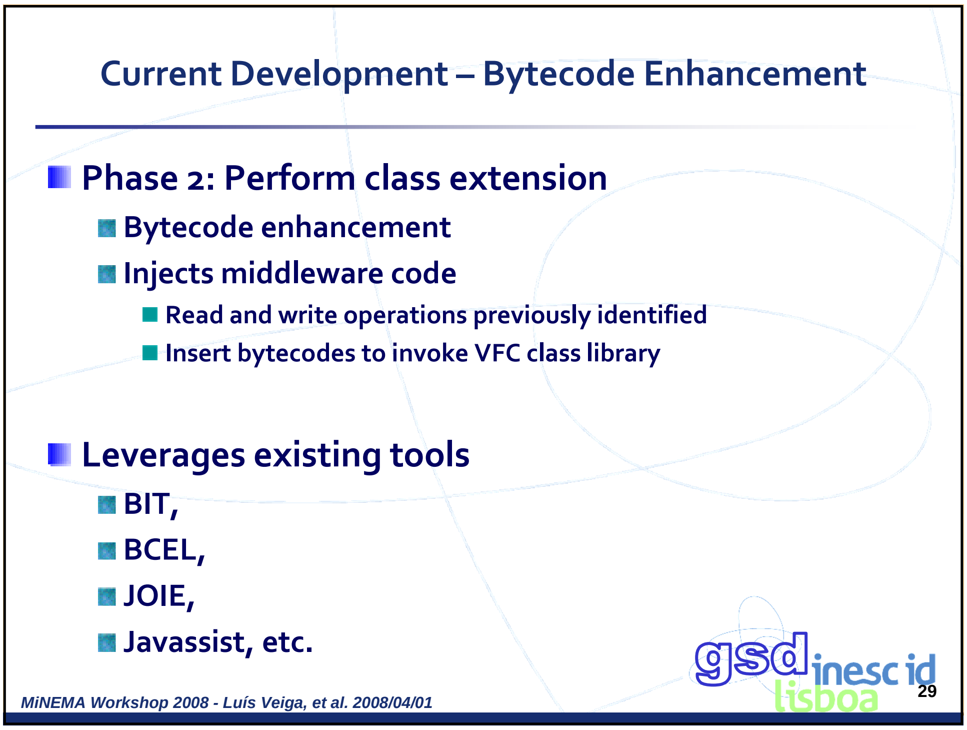
Talk Outline

- *…Introduction*
  - Motivation
  - Problems and Current Approaches
  - Insights and Basic Idea
- *…Vector-Field Consistency…*
  - Approach, Model, Entities, Architecture
  - Mobihoc: VFC for Ad-hoc Gaming
  - Evaluation and Language Integration
- ***Beyond…***
  - Current Development
  - On-going and Future Work



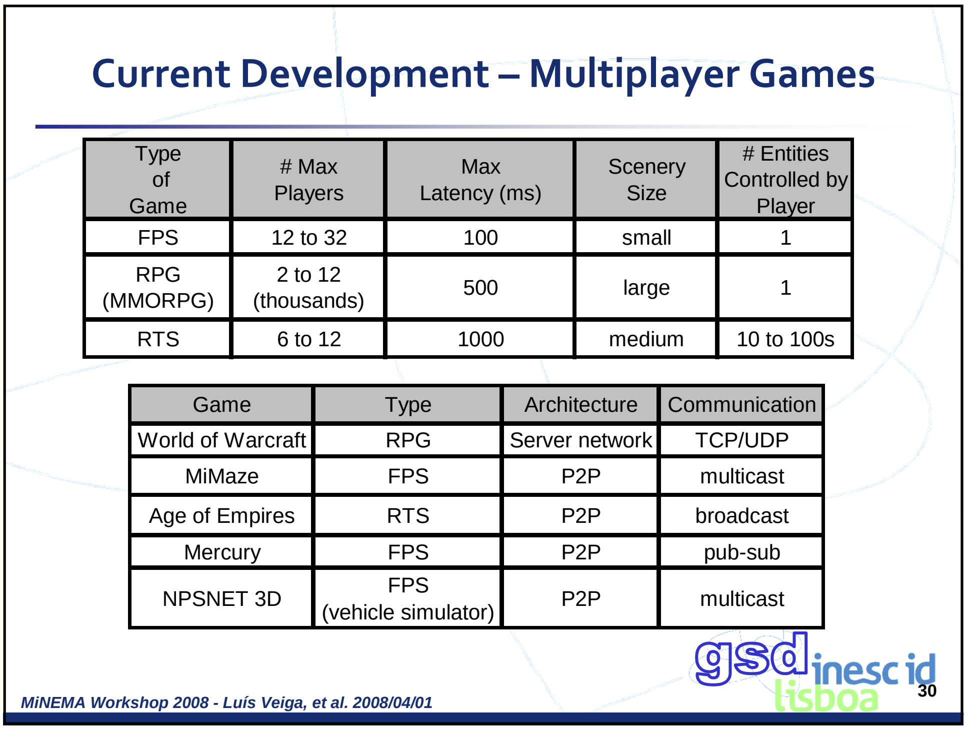
Current Development – Bytecode Enhancement

- To apply VFC when source code unavailable
  - Extracts availabe information from:
  - annotations/attributes/comments
- Phase 1: Analysis
  - Processes XML files with additional information:
    - relevant object field names
    - e.g. player position, etc.
  - Analyses class byte-codes
    - looks for operations on replicated data
    - reads and updates to relevant fields
    - annotates metods with Read/Write tags and places custom op-codes



Current Development – Bytecode Enhancement

- Phase 2: Perform class extension
  - Bytecode enhancement
  - Injects middleware code
    - Read and write operations previously identified
    - Insert bytecodes to invoke VFC class library
- Leverages existing tools
  - BIT,
  - BCEL,
  - JOIE,
  - Javassist, etc.



Current Development – Multiplayer Games

Type of Game	# Max Players	Max Latency (ms)	Scenery Size	# Entities Controlled by Player
FPS	12 to 32	100	small	1
RPG (MMORPG)	2 to 12 (thousands)	500	large	1
RTS	6 to 12	1000	medium	10 to 100s

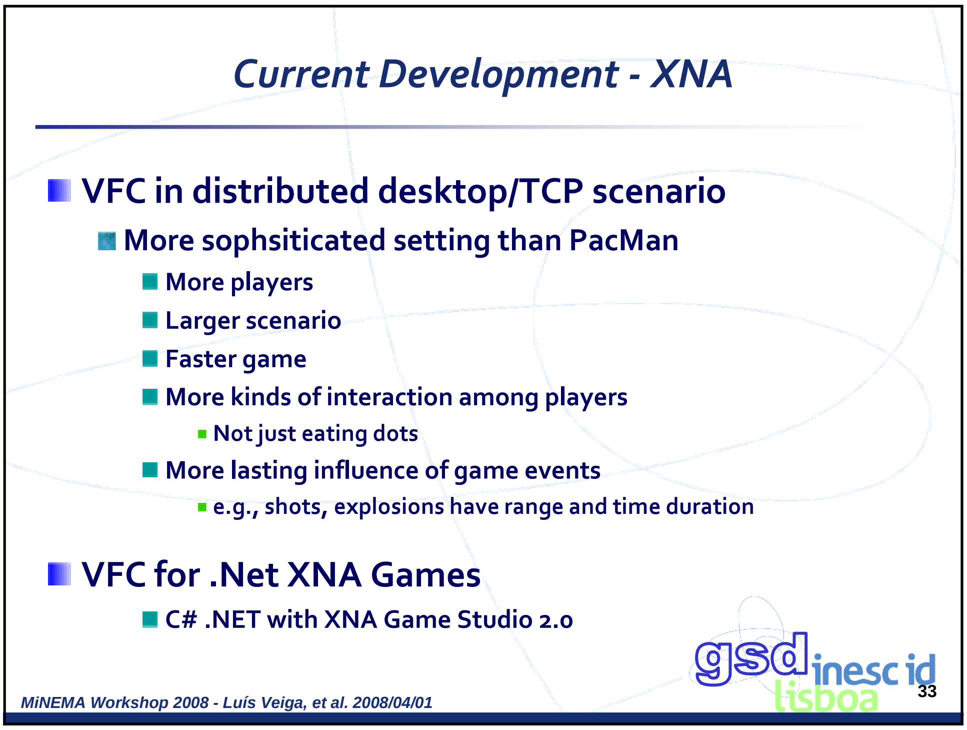
Game	Type	Architecture	Communication
World of Warcraft	RPG	Server network	TCP/UDP
MiMaze	FPS	P2P	multicast
Age of Empires	RTS	P2P	broadcast
Mercury	FPS	P2P	pub-sub
NPSNET 3D	FPS (vehicle simulator)	P2P	multicast

Current Development – Multiplayer Games

- VFC better suited for FPS
  - Actually designed for them
  - Easier to identify *pivots*, and define *consistency zone radius*
  - Also applies to RPG but less demanding
- VFC less suited to RTS
  - User controls large number of entities
  - No single spatial position for each user
  - More difficult to identify pivots
  - More pivots to manage
    - hopefully, will imply smaller consistency zones

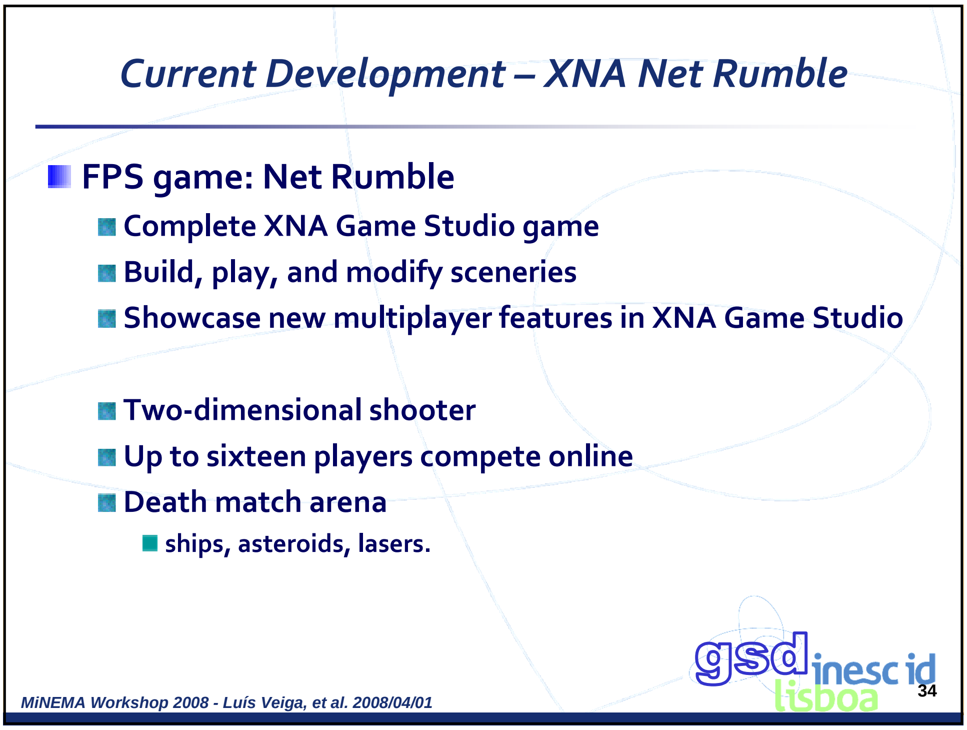
Current Development – Multiplayer Games

- VFC better suited for FPS
  - Actually desgined for them
  - Easier to identify *pivots*, and define *consistency zone radius*
  - Also applies to RPG but less demanding
- VFC less suited to RTS
  - User controls large number of entities
  - No single spatial position for each user
  - More difficult to identify pivots
  - More pivots to manage
    - hopefully, will imply smaller consistency zones



Current Development - XNA

- VFC in distributed desktop/TCP scenario
  - More sophsiticated setting than PacMan
    - More players
    - Larger scenario
    - Faster game
    - More kinds of interaction among players
      - Not just eating dots
    - More lasting influence of game events
      - e.g., shots, explosions have range and time duration
- VFC for .Net XNA Games
  - C# .NET with XNA Game Studio 2.0

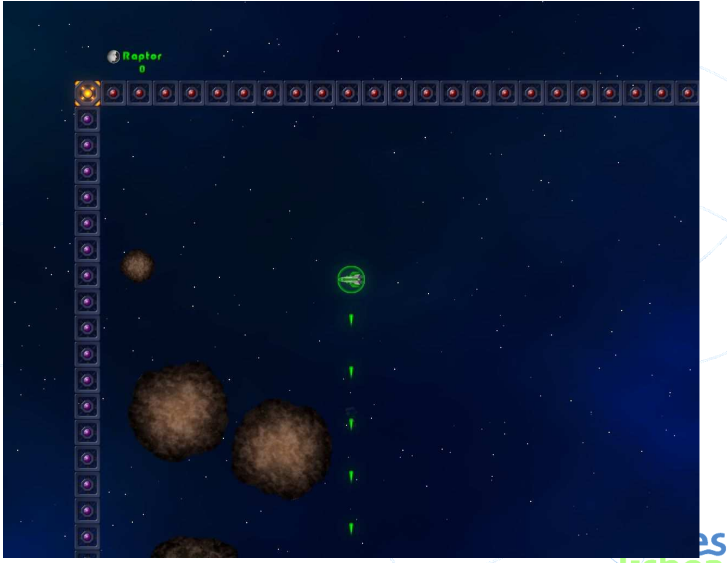


Current Development – XNA Net Rumble

- FPS game: Net Rumble
  - Complete XNA Game Studio game
  - Build, play, and modify sceneries
  - Showcase new multiplayer features in XNA Game Studio

  - Two-dimensional shooter
  - Up to sixteen players compete online
  - Death match arena
    - ships, asteroids, lasers.

XNA Net Rumble



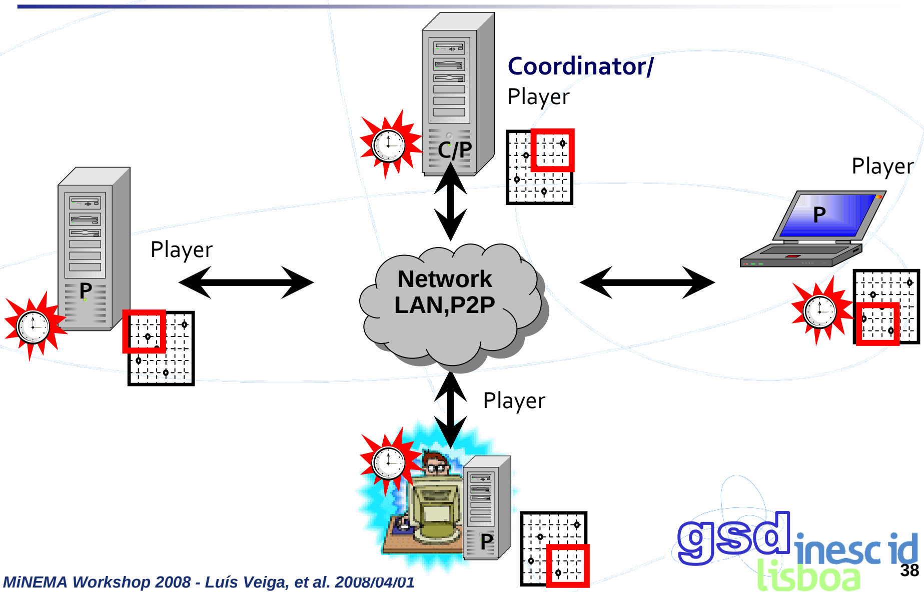
Current Development – XNA Net Rumble

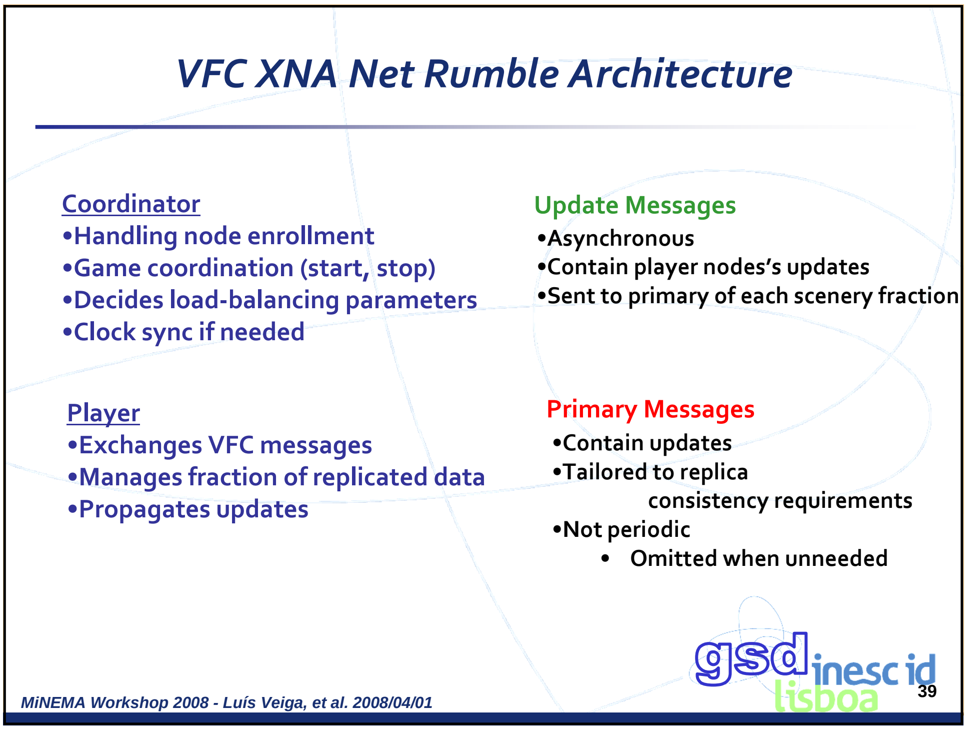
- Overall Approach
  - Impact: Measure source code modifications
    - Favouring declarative and autogenerated source code extensions
    - *Partial class* feature in C# from 2.0
  - Evaluation:
    - confirme game rules compliance inspite of divergence bounding
  - Performance:
    - measure traffic reduction
      - different scenarios (zones, players, etc)
    - measure CPU load

Current Development – XNA Net Rumble

- Architecture
  - Centralized (Client-Server)
    - Simplified development
    - Game rules verification
    - Reference for performance measurements
  - Distributed (P2P)
    - Increased availability and performance
    - Larger number of players
    - Potentially increase VFC benefits
    - Still on-going work
      - Currently employs one peer as coordinator

VFC XNA Net Rumble Architecture





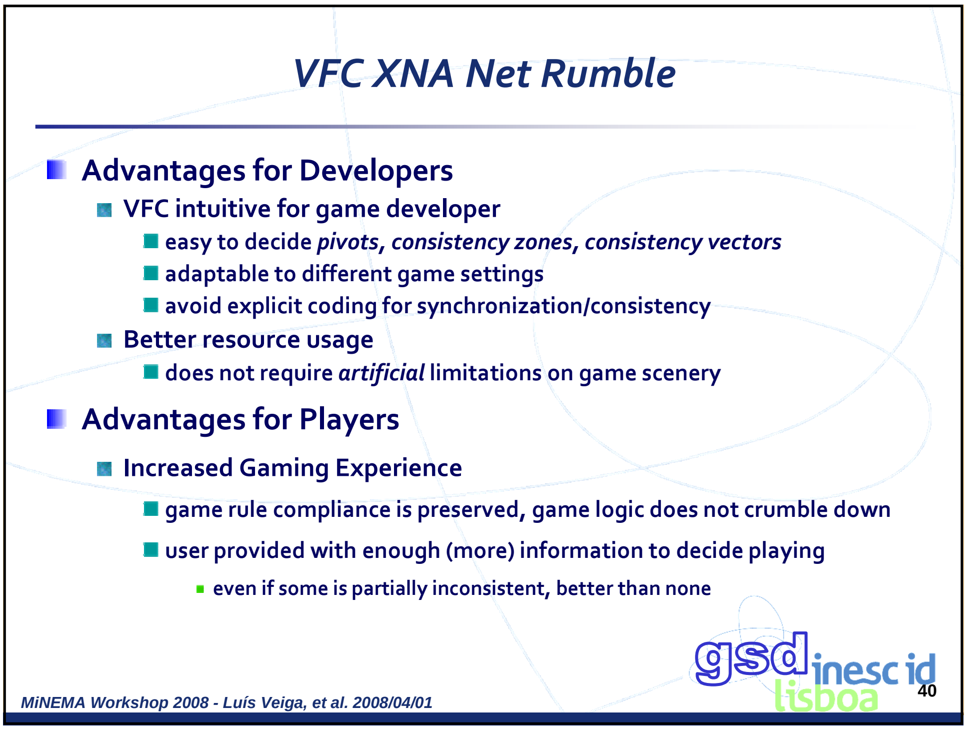
VFC XNA Net Rumble Architecture

Coordinator
- Handling node enrollment
- Game coordination (start, stop)
- Decides load-balancing parameters
- Clock sync if needed

Player
- Exchanges VFC messages
- Manages fraction of replicated data
- Propagates updates

Update Messages
- Asynchronous
- Contain player nodes's updates
- Sent to primary of each scenery fraction

Primary Messages
- Contain updates
- Tailored to replica consistency requirements
- Not periodic
  - Omitted when unneeded

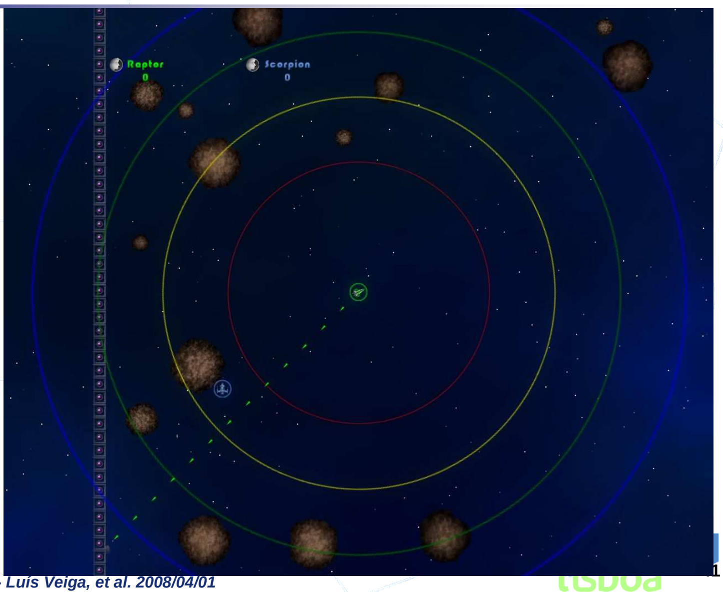


VFC XNA Net Rumble

- Advantages for Developers
  - VFC intuitive for game developer
    - easy to decide *pivots*, *consistency zones*, *consistency vectors*
    - adaptable to different game settings
    - avoid explicit coding for synchronization/consistency
  - Better resource usage
    - does not require *artificial* limitations on game scenery
- Advantages for Players
  - Increased Gaming Experience
    - game rule compliance is preserved, game logic does not crumble down
    - user provided with enough (more) information to decide playing
      - even if some is partially inconsistent, better than none

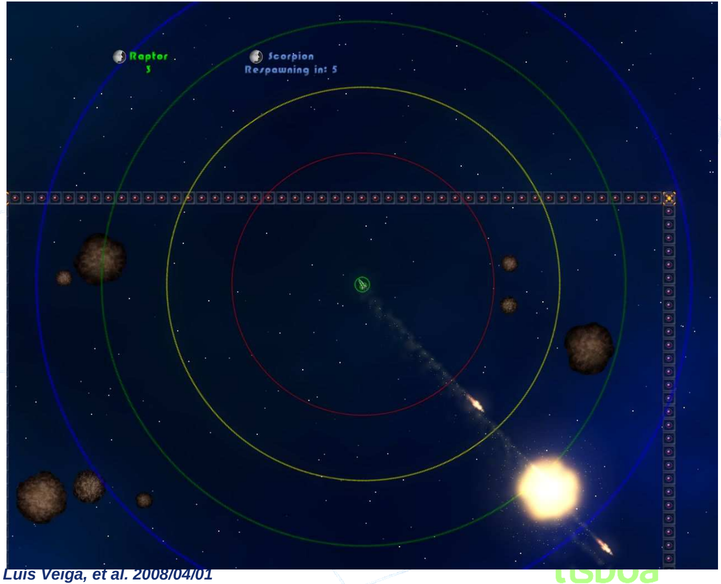
VFC XNA Net Rumble

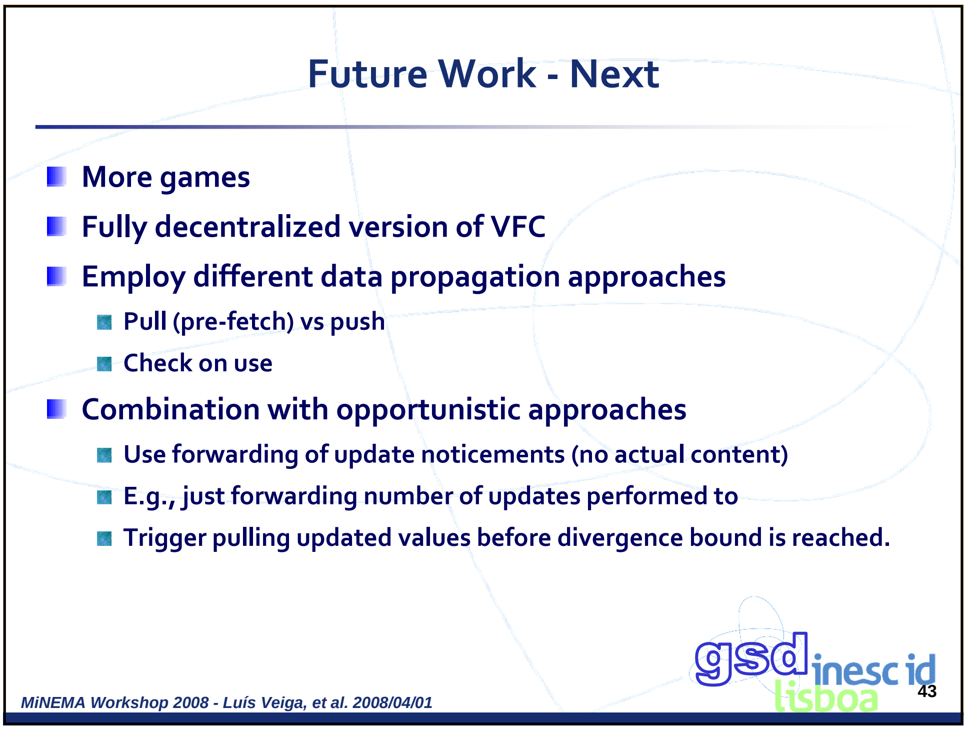
- Larger Scenery
- Wider Visualization
- Consistency Zones



XNA Net Rumble with VFC

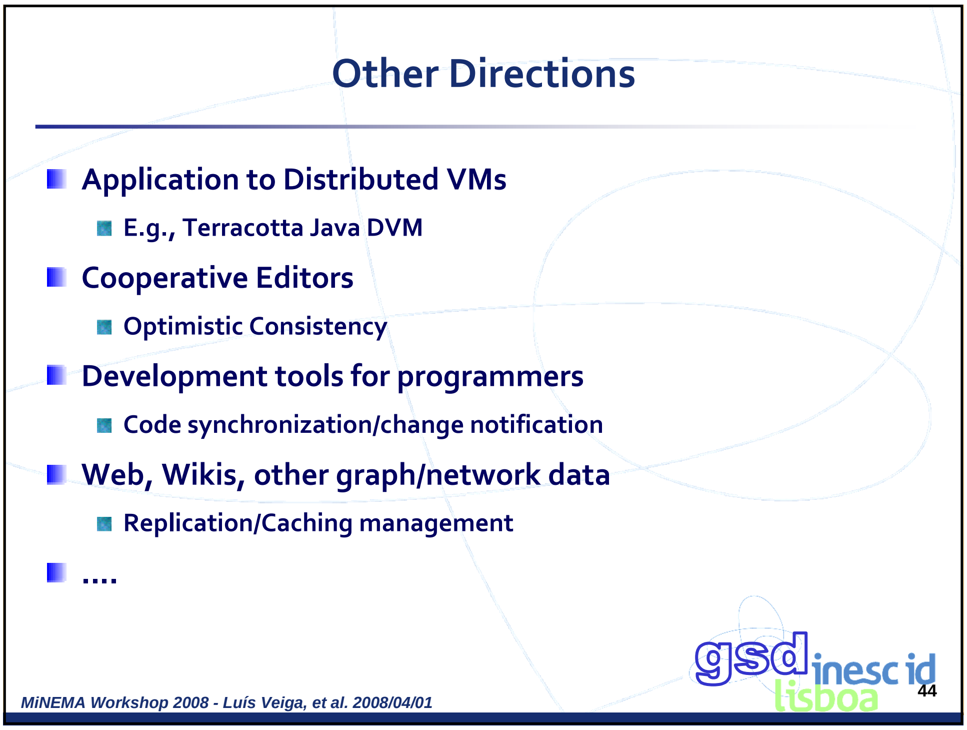
Shooting
And
Explosion





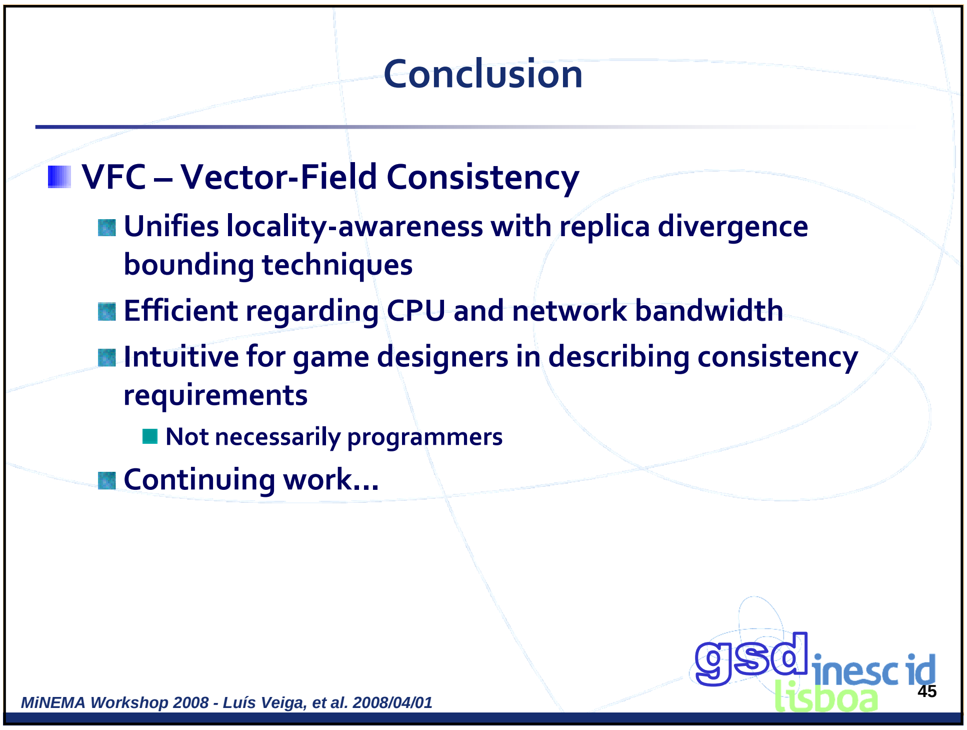
Future Work - Next

- More games
- Fully decentralized version of VFC
- Employ different data propagation approaches
  - Pull (pre-fetch) vs push
  - Check on use
- Combination with opportunistic approaches
  - Use forwarding of update noticements (no actual content)
  - E.g., just forwarding number of updates performed to
  - Trigger pulling updated values before divergence bound is reached.



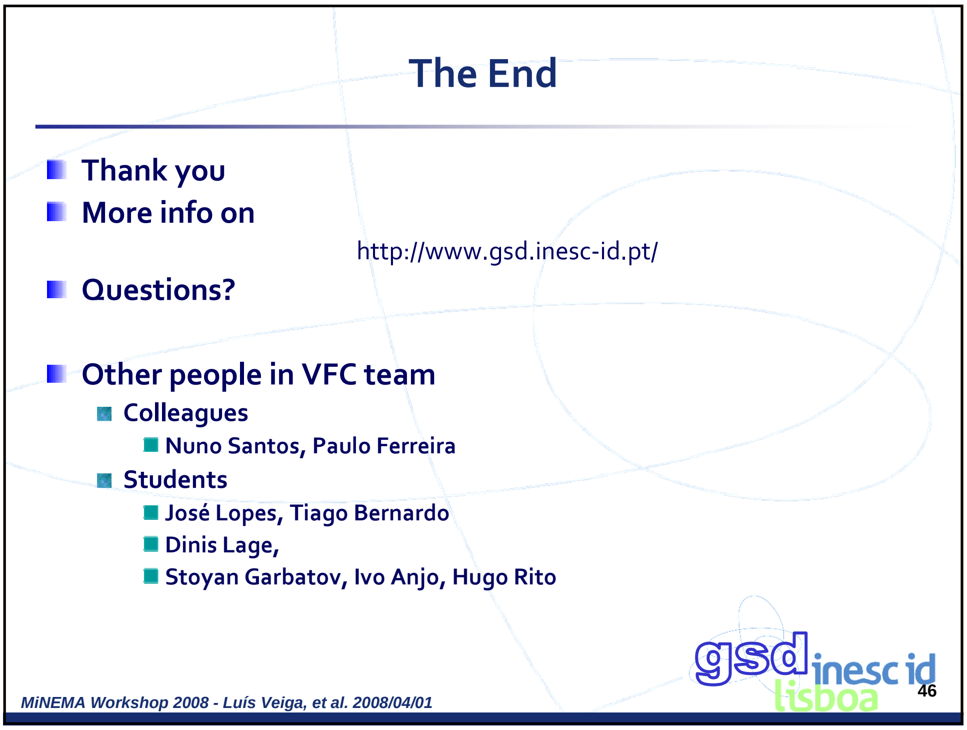
Other Directions

- Application to Distributed VMs
  - E.g., Terracotta Java DVM
- Cooperative Editors
  - Optimistic Consistency
- Development tools for programmers
  - Code synchronization/change notification
- Web, Wikis, other graph/network data
  - Replication/Caching management
- ….



Conclusion

- VFC – Vector-Field Consistency
  - Unifies locality-awareness with replica divergence bounding techniques
  - Efficient regarding CPU and network bandwidth
  - Intuitive for game designers in describing consistency requirements
    - Not necessarily programmers
  - Continuing work…



The End

- Thank you
- More info on

http://www.gsd.inesc-id.pt/

- Questions?

- Other people in VFC team
  - Colleagues
    - Nuno Santos, Paulo Ferreira
  - Students
    - José Lopes, Tiago Bernardo
    - Dinis Lage,
    - Stoyan Garbatov, Ivo Anjo, Hugo Rito