Definition of Application Scenarios

G. Biegel, G. Blair, V. Cahill, A. Casimiro, K. Cheverst,
R. Cunningham, A. Fitzpatrick, A. Friday, G. Gaertner,
B. Hughes, J. Kaiser, R. Meier, N. Riejers and P. Veríssimo

DI-FCUL TR–03–14

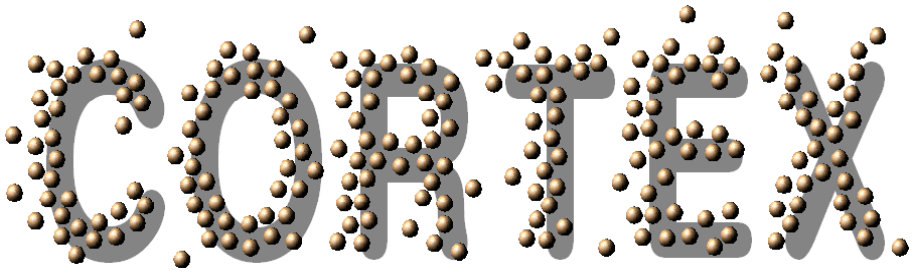
July 2003

Departamento de Informática
Faculdade de Ciências da Universidade de Lisboa
Campo Grande, 1700 Lisboa
Portugal

Technical reports are available at http://www.di.fc.ul.pt/tech-reports. The files are stored in PDF, with the report number as filename. Alternatively, reports are available by post from the above address.

Project IST-2000-26031

CO-operating Real-time senTient objects:
architecture and EXperimental evaluation



Definition of Application Scenarios

CORTEX Deliverable D1

Version 1.0

October 31, 2001

Revisions

Rev.	Date	Comment
0.1	21/10/2001	Integration of partner contributions
0.2	30/10/2001	Integration of updates from TCD and Ulm
0.3	31/10/2001	Integration of updates from Lancaster
0.4	06/11/2001	Proofreading and cross reference checking
1.0		Final version

Editor

Vinny Cahill, Trinity College Dublin

Contributors

Greg Biegel, Trinity College Dublin
Gordon Blair, University of Lancaster
Vinny Cahill, Trinity College Dublin
Keith Cheverst, University of Lancaster
Antonio Casimiro Costa, University of Lisbon
Raymond Cunningham, Trinity College Dublin
Aidan Fitzpatrick, Trinity College Dublin
Adrian Friday, University of Lancaster
Gregor Gaertner, Trinity College Dublin
Barbara Hughes, Trinity College Dublin
Jörg Kaiser, University of Ulm
René Meier, Trinity College Dublin
Neils Riejers, Trinity College Dublin
Paulo Jorge Verissimo, University of Lisbon

Address

Department of Computer Science,
Trinity College Dublin,
Ireland

Table of contents

Chapter 1 Introduction

As described in [1], the purpose of this document is to identify and analyze a set of application scenarios that, on the one hand, exemplify those application areas that might benefit from the technology being developed within the CORTEX project and, on the other hand, might serve as a source of requirements on this technology. Furthermore, at least a subset of the application scenarios considered here is expected to serve as source of demonstrator applications later in the project.

We proceed as follows. Chapter 2 provides a brief overview of the most important properties that are expected to be exhibited by the applications addressed by CORTEX, as outlined in [1], and introduces the scheme that we have adopted for classifying candidate applications in this document. The classification scheme essentially considers applications under three key characteristics: 1) autonomy - the degree to which the participants in the application act solely based on the acquisition of information from the environment and on their own knowledge; 2) consistency - the degree to which participants in the application need to have a mutually consistent view of the application environment; and 3) cooperation, the degree to which the participants in the application need to cooperate in order to achieve their goals. Chapter 3 describes the analysis of a number of proposed application scenarios in detail outlining the scope of each scenario, how it has been classified with respect to the other scenarios and, of course, the specific requirements that it places on the CORTEX technology. For each scenario, the feasibility of basing a demonstrator application on that scenario is also considered. Chapter 4 summarizes the results of this survey.

Chapter 2 Characteristics of Application Scenarios

Fundamentally, CORTEX is concerned with enabling the vision of ubiquitous computing and future proactive applications, those that operate independently of direct human control, by researching the development of intelligent middleware supporting appropriate computational models for this new generation of applications. This middleware must support growth and adaptability to new technologies, and has to provide the hooks for these applications to enforce non-functional quality attributes like reliability and timeliness. In particular, the middleware is intended to cope with applications that have *some or all* of the following characteristics:

- Sentience – the ability to perceive the state of the surrounding environment, through the fusion and interpretation of information from possibly diverse sensors;
- Autonomy – components of these applications will be capable of acting in a decentralized fashion, based solely on the acquisition of information from the environment and on their own knowledge;
- Large scale - typical applications may be composed of billions of interacting hardware and software components;
- Time criticality - these applications will typically interact with the physical environment, and will have to cope with its pace, regardless of adverse conditions due to scale and technology shortcomings;
- Safety criticality – typical applications will interact with human users, whose well being will frequently rely on them;
- Geographical dispersion - unlike current embedded systems, typical applications will integrate components that are scattered over buildings, cities, countries, and continents;
- Mobility – furthermore, they must possess the ability to move between hosts possibly of different networks, while remaining in continuous operation
- Evolution – these applications will have to cope with changing conditions during their lifetimes. Not only must the applications be designed to evolve, but their underlying support must also be adaptable.

It is unlikely that every or indeed any one application will exhibit all of these characteristics at the same time. Indeed, it can be expected that most applications will exhibit only some of these characteristics. What seems clear is that the most challenging applications are those that mix autonomy of application participants, derived from sentience, with the need to maintain a consistency view of the application environment while possibly cooperating with other participants.

Autonomy is a central feature of the participants in the kind of proactive applications being targeted by CORTEX and, in some sense, the motivation for sentience, which provides autonomous components with the knowledge required for independent action. While there is clearly a spectrum of levels of autonomy, CORTEX is clearly targeted at applications where some degree of autonomous behavior is required.

While sentience provides participants in an application with knowledge of the environment it is clearly important that the view obtained by the participant is consistent not only with the views of other participants but also with the real world. Consistency then is another key feature of the applications addressed by CORTEX, which in some sense subsumes requirements such as timeliness.

Consistency is also the basis for cooperation between autonomous components. While not all applications will involve explicit cooperation in order to achieve application goals,

cooperation will often be required implicitly to ensure properties such as safety and at a system level to priories the use of computational resources.

With these observations in mind, the CORTEX project has used a three-way classification of candidate application scenarios to expose the key differences between the scenarios being considered as well as to ensure that the candidate scenarios cover a wide range of usage scenarios.

To be precise, the classification scheme considers applications under three characteristics:

1) autonomy, the degree to which the participants in the application act solely based on the acquisition of information from the environment and on their own knowledge
2) consistency, the degree to which participants in the application need to have a (mutually) consistent view of the application environment; and
3) cooperation, the degree to which participants in the application need to cooperate in order to achieve their goals.

Each candidate application scenario is considered under these headings in Chapter 3 and the applications compared with respect to these headings in Chapter 4.

Chapter 3 Application Scenarios

3.1 Flare Scenario

“Shoot 'em up” style games like Doom and Quake have become very popular in recent years. Paintball, a real world outdoor “Shoot 'em up” game using paint-filled pellets as bullets, and Lasergame, an indoor game, which uses laser beams and sensors on the players' suits, have also become very popular.

Flare is a framework for building augmented reality applications, the first application of which will be a Doom-like game combining the Doom/Quake experience with Paintball/Lasergame play to make an augmented reality game where players move around in the real world, while interacting with virtual and real players. Players will see a Doom-like game on their screen, providing a virtual representation of the real world. Their real world position will determine their location in the game. There will be both real players and virtual, computer controlled, bots in the game. Players can shoot bots and other players and pick up things like ammunition or medikits as in a normal Doom game. Players will be able to join and leave the game as they please, but the game will cease to exist when the last player leaves.

Flare will be run on wearable computers. A wireless ad hoc network will be used for communication, and Differential Global Positioning System (DGPS) and possibly other sensors to determine location. Flare will be developed in three stages. The first version will use group communication and support fault- and partition-tolerance using a single group for the entire game. Players will communicate through the group, broadcasting information like their new position, or the fact that they have fired their guns. Nodes will only update the game state as a result of receiving a message. The group communication API will support non-blocking communication even when the network is partitioned, delivering messages to the partition instead of the whole group. The receiving nodes are notified of this, and can respond to this to maintain consistency.

The second version of Flare will use multiple groups. The game area will be split into multiple zones and different groups will be used for different zones. Different groups will also be used for different interests. For instance, all bots and players in a team could use a team group to coordinate their attack. Thus, the second version will have filtering using multiple groups based on both location and interest. The third version will extend Flare with event-based communication.

Zones map nicely onto the second version of Flare where multiple groups are used for different geographical locations. These can be seen as zones, which, because of there smaller geographical size, can give higher quality of service (QoS) guarantees than would be possible if the whole game was one big zone. Basically players and bots within a zone play the game almost separately from the rest of the players, and will only require communication with the others for important events.

3.1.1 Consistency

Since this is a game application, users are bound to disconnect suddenly if they get bored with the game. The use of mobile computers and wireless networks also makes failures and network partitions likely. To allow the game to progress as much as possible in this environment, the game state will be replicated on all nodes, and the game will use producer/consumer communication instead of the client-server model that is common for these games. A client-server model would be unsuitable because nodes that lose contact with the server cannot make progress and if the server failed the whole game would stop.

Having replicated data means that we need to formulate consistency requirements. The easiest choice would be to require all nodes to have a consistent view of the game, but the game would have to block if the network is partitioned. Since we want to make progress in the presence of partitions as well, we define different levels of consistency.

We will have objects in the game, like medikits, for which we don't mind if the state becomes inconsistent when a partition occurs. This means that two players can potentially pick up the same kit. We will have objects for which we will allow one partition to do updates, and will simply copy that to the other when partitions remerge. For example, in a "capture the flag" type game, we cannot allow two players to pick up the same flag. Therefore we would allow only one partition to change the state of the flag, so they can pick it up. Deciding which partition can change the state would be done using location information, so partitions would be allowed to change the state of objects that are close to them, which will hopefully minimize the negative effects of the partition. Finally, we will have state information for which it is absolutely required that all nodes have the same value. For example, deciding who won the game. Since this will end the game, we want all players to agree on a common winner, and so no winner can be elected if the network is partitioned.

3.1.2 Autonomy

Obviously, the players in Flare are autonomous and will make decisions based both on their perceptions of the real world and the virtual world. The bots will also operate autonomously in the game under computer control although their actions are fairly limited. Although their state is fully replicated, there will be one node that is responsible for initiating the bot's actions. Deciding on which node this is will probably be done using location information to keep the physical location of the node as close to the virtual location of the bot as possible. A bot will have to autonomously decide that it wants to move to another node and initiate the appropriate protocol to do this.

3.1.3 Cooperation

Cooperation is not very important in Flare, but the game rules could be modified to include teams fighting each other. In this case, the bots and players would need to cooperate to win the game. The goals can be changed to make this more challenging. An example would be two teams that have to shoot all members on the other team. But the goal could also be to plant a bomb, to rescue hostages or to capture a flag. For these goals the level of cooperation and the complexity of the autonomous behavior would be much higher.

3.1.4 Requirements Analysis

3.1.4.1 Requirements from the environment

Flare does not address:

- Large number of physical entities: flare will be played with a small number of players.
- High geographical dispersion of entities: the players will play in a limited game space that has been modeled as a virtual world.
- Human safety relies on computer-controlled entities: playing the game is unlikely to put the physical well being of the participants at risk.

Flare does address the following:

- Computer control of entities requires timely response: in order to make sure that what happens in the game is consistent with the chain of events observed in the real world, there will be timeliness requirements for message delivery.

- Subset of entities should be highly mobile: the players will move around quickly in the real world.
- State of entities changes rapidly: players shooting and dying, things being picked up and dropped.
- Feasible to demonstrate: the infrastructure necessary to implement such a system is available.
- Relevance for society/industry: the popularity of these types of games proves they are relevant to society.

3.1.4.2 Requirements from the computer system

Flare does not cover:

- Support for scalability: the game will be played with a limited number of players only, scalability is not an issue.
- Support for real time computing enforcing hard QoS requirements: there are quite demanding timeliness requirements on message but failing to deliver a message in time is not catastrophic.
- Support for adaptability/evolution of the system: the system only evolves in the number of players.
- Support for heterogeneity and openness: the game is essentially closed although the use of different game devices can be envisaged.

Flare does cover:

- Support for fault-tolerance: the game should react in a consistent and well-defined way to partitions and failures while still allowing players to make as much progress as possible.
- Support for autonomy: both, the players and the bots will behave autonomously.
- Support for sentience: both, in the real world through location sensors and in the virtual world for the bots.
- Support for mobility with wireless communication: the game will be played on wearable computers with wireless 802.11 network cards.

3.1.4.3 Techniques/mechanisms

Not useful in Flare:

- Real-time networks
- Actuator technology

Useful in Flare:

- Filtering: a version of Flare will use filtering of messages both geographically and functionally.
- Group communication: only one group will be used in the first version, but multiple groups in later versions.
- Event-based communication.
- Cooperation mechanisms.
- Sensor technology: different position sensors will be used to determine the player's location.
- Wireless networks.

3.1.5 Feasibility

Perhaps the main reason why Flare is an interesting application scenario is because of its feasibility. It could be implemented in a relatively short period of time, and it can be run using inexpensive hardware. Another advantage of using Flare is that because it is a game, we can chose the rules in such a way as to address the issues in which we are interested. They are not fixed by real world requirements, so we can make things as easy or as hard as we like and explore different directions.

3.2 Air Traffic Control Scenario

CORTEX aims to provide sentient objects - mobile intelligent agents, which accept input from a variety of different sensors and reacting to what they have sensed from the environment. One of the main challenges of the CORTEX project is to ensure timeliness and predictability when confronted with a dynamic environment.

In the Air Traffic Control (ATC) scenario, the sentient objects are the aircraft. The main reason that we consider ATC as a possible CORTEX application is that the central goal of CORTEX, to provide completely autonomous, decentralized cooperating agents, must be met in ATC or the results will be catastrophic. The ATC scenario requires a high degree of autonomous behavior, strong consistency and a high degree of cooperation among the sentient objects.

The ATC scenario is envisaged as providing a "free flight" system. Each aircraft will be equipped with sensing equipment to detect other aircraft in the immediate airspace. This could be thought of in terms of the WAN-of-CANs type architecture. Each aircraft has an internal CAN architecture, which is then controlled over a wider airspace. As stated above, the ATC scenario has requirements for high consistency, high autonomy and cooperation. Each of these will be further discussed in the following sections.

3.2.1 Consistency

In the ATC scenario aircraft will share airspace, and sense their immediate environment to determine the existence of other aircraft. Changes of status (e.g., changing speed, direction, altitude, etc.) of the other aircraft must be constantly monitored. The only way that an aircraft can make a judgment as to their "next move" is if each is guaranteed an up-to-date snapshot of their immediate vicinity. This snapshot must have a timeliness and correctness guarantee. Craft share a common environment; therefore it must be ensured that all craft share a common view as to the state of the environment.

This consistency requirement is paramount for the operation of a free flight scenario, and without it the results would be catastrophic. The following real-world scenarios highlight this requirement.

Monitoring airspace and "special rules airspace"
At present, there is a requirement to continually monitor the airspace that an aircraft is currently in. The current airspace is a moving target, i.e. continually updated throughout the flight and is subdivided into groups. En route, the aircraft moves between groups. An important point is that there are "special rules airspaces" which includes standard transport craft and military craft.

For free flight there would need to be continuous snapshots of the current airspace, consistently updated to all those in the proximity of the current aircraft. Proximity detection, to determine the current positioning of other craft, coupled with high-speed sensor equipment to ensure high consistency is required.

A question about the implications of the military craft in the same airspace as normal aircraft would have to be raised. Would they participate in the same form of groupings? We have assumed that free flight will exist for standard flights and special conditions will use the

existing Air Traffic Control system. If free flight means that there is no longer ground control interaction, and aircraft detection is based on sensing an object in the immediate environment, the idea of airspace as an entity is no longer required. An aircraft will monitor its position with respect to all others that it can detect.

Monitoring separation standards for differing aircraft types
We are assuming that aircraft have sensory equipment such that they can sense other aircraft within a large area, this is similar to the current use of radar controllers monitoring the current airspace of the craft, throughout the course of a flight. There are currently separation standards in place for both vertical and horizontal distance between planes. The distance must take into account the size, load and type of aircraft, for free flight; these considerations must also be taken into account. The direct implication of this is that each craft must be able to determine proximity but also aircraft type, and use all these parameters to determine a safe separation distance.

Handling "over flying"
A concept of over flying, which for current ATC means the craft are not actually in a dedicated airspace, exists. This would be directly handled by removing the reliance on defined airspaces, and passing control between the ATC handling each airspace. As stated above, we are assuming that the traditional airspace concept no longer exists, over flying is therefore not an issue.

Hard real-time timeliness requirements
There are time-critical consistency requirements for all aircraft within a defined distance of one another. They must always have a current snapshot of the other participants of their group. Each aircraft has a high level of autonomy, as realistically they can plot their own course, but essentially, as will be described, this is effectively within agreement from the other craft in their group. A group may be taken as the set of other aircraft that are currently detectable.

We are proposing a protocol whereby a plane wishing to change its course (or to do any operation that may have an impact on others) sends out a multicast message to all aircraft that it knows about. This is analogous to dynamic membership in group communications. There must be some mechanism to ensure that all the planes affected agree that the change of course can be taken. Essentially, the initiator must block awaiting a consensus on the proposed change. Only when this is achieved should the change take place.

What happens in the case of failed aircraft, i.e., an aircraft that does not receive the multicast message. Essentially, the initiator must hold off making a move until a reply has come from all the participants that are affected by the move. This could lead to deadlock where no plane can actually make progress. Absolutely hazardous, no plane could change its direction, altitude, or speed.

3.2.2 Autonomy and Cooperation

Free flight implies completely autonomous aircraft. The days of filing flight plans, negotiating with control towers for such things as route congestion avoidance have been left far behind. The aircraft now have complete control from start to finish. This obviously provides a high level of autonomy, however there are occasions when cooperative behaviour is required. For example, any changes to aircraft course must be broadcast to all those who may be affected, take-off and landing require a large amount of negotiation and cooperation and are dealt with specifically in sections following.

Handling route congestion
ATC, at present, attempts to remove route congestion. For free flight systems, this would have to mirror the Assisted Terrestrial Transportation System (ATTS) scenario (see section 3.4). Essentially if one craft can determine that the current route is overly congested, it can propagate this information to others within the group, which may mean that they leave the current group and join another. Essentially meaning that the other craft have decided to change their flight path.

Special Landing requirements
Special consideration must be taken at landing. Essentially there are defined "holding stacks" where aircrafts wishing to land must circle. Essentially an aircraft must negotiate with others if/when it is permitted to land. We consider the landing area as a priority queue, where each aircraft requesting to land broadcasts to all those in the queue, their current status. For example, the amount of fuel they have left or any on-board conditions that must be taken into account when deciding their position in the holding queue. Each aircraft could be given an initial priority, which may require rapid update if there are any changes to the state of the aircraft. The consistency of the information for all the craft in the holding queue is paramount, and their cooperation in deciding the outcome should a queue reorganization be requested is essential. Situations where there may be a race for a particular slot in the queue must be fairly arbitrated. How would fairness be guaranteed throughout the queue? This scenario highlights the requirements for cooperation amongst the participating planes.

Special considerations for take-off
The take-off scenario is similar to the above. Again, there must be some form of priority queuing mechanism to ensure fairness amongst all craft awaiting departure. The constraints on the queue may be simplified as there may not be a requirement to maintain the priority of the queue, this could be a FIFO ordered queue instead. Take-off is similar to the scenario above, where a highly cooperative group structure is required amongst the planes.

Handling international standards/boundaries
Some further thoughts would be required on areas such as: International standards, boundaries, sectors and sector controllers. For example, at present craft from some countries are prohibited from flying over others. The "flight plan" defined for the aircraft would stringently avoid these areas. If free flight is to include total dynamic route management, this implies craft are basing their decisions on such information as weather conditions, congestion etc. in an attempt to traverse the optimal path for the current destination. In addition to these factors, information on boundary control etc. would have to be taken into consideration, implying that there must be some way that the craft can sense when it is passing from a free sector to what would be a restricted sector. Similar to ATTS the first aircraft to detect this situation can propagate this information to all others, removing the requirement that each craft independently find out and react to this situation.

To summarize autonomous and cooperative behavior
The level of autonomy of the aircraft would seem to increase the further they move away from the airport. There is a large amount of negotiation that needs be performed when requesting to take off, and requesting to land, and all other craft within the vicinity of the airport, both grounded and airborne, are involved in these negotiations. En route, the aircraft within a defined proximity of each other are involved in negotiation only.

All aircraft must have a completely consistent view of all other aircraft in their vicinity. We have assumed that this view is maintained by sensing the other craft. Alternatively, an updated view is sent to a known group, the membership of which is dynamically updated, along communication channels. For an aircraft to perform any sort of maneuver, it must receive a consensus from the other members in its vicinity (or in the group). The implication of this is that there is a high level of cooperation required for an aircraft to make progress.

3.2.3 Requirements Analysis

3.2.3.1 Requirements from the environment

ATC does address the following:

- A dynamic and potentially very large number of physical entities.
- Hard timeliness requirements: there are real world considerations in ATC, whereby a missed deadline has catastrophic results
- The state of the entities may change rapidly: any unforeseen event, such as engine failure, will require an immediate course of action to be taken for the new state.
- The idea of free flight has been gaining momentum in recent times, therefore the ATC scenario is gaining popularity.

3.2.3.2 Requirements from the computer system

ATC would have to cover:

- Support for autonomy.
- Support for scalability: the dynamics of the environment are ever changing. Essentially there is no upper limit on the planes involved.
- Support for time dependent real time computing enforcing hard QoS requirements.
- Support for adaptability/evolution of the system: the ATC would have to evolve with changes in the aviation industry.
- Support for fault-tolerance: at all times a fault in the system implies a huge cost and potential loss of life.

3.2.3.3 Techniques/mechanisms

Useful in ATC:

- Real-time networks
- Anonymous communication
- Actuator technology
- Group communication
- Event-based communication

3.3 Utilities Industry Scenario

In order to provide electricity to its consumers the power distribution industry has to manage a mass of complex cabling and each regional electricity company (REC) has to deal with managing the supply of electricity to approximately 1.8 million electricity consumers. Problems with the power distribution arise quite frequently and when such problems occur and consumers are left without supply there is strong financial incentive for the REC to return supply as soon as possible. Usually when a fault occurs on the network it is necessary to re-route supply via an alternative path, this is known as ‘switching’. Field engineers are required to physically perform switching operations at ‘Switching stations’. In the current system, a control centre is responsible for coordinating the work of field engineers and for maintaining an up-to-date view of the network state. For reasons of safety, it is crucial that such a view be maintained and that no inconsistencies regarding the current network state exist. In order to maintain the consistency of network views the control centre imposes a sequential ordering on all operations affecting the network.

The current centralised approach has the inherent problem that during periods of intensive activity, e.g. during an electrical storm, the control centre can become a system bottleneck. This situation is clearly not acceptable, and a less centralised control structure needs to be put in place. Ideally, this would be achieved by distributing (i.e. replicating) the current network view amongst all concerned, rather than permitting the control centre to hoard the current view. There are obviously strong implications for maintaining consistency where multiple copies of the current network view exist. These implications are made all the more dramatic when one considers the unreliable communications infrastructure available for sending updates between the control centre and mobile field engineers.

Our approach with the MOST project was based on distributed object technology (ansaWARE) augmented by groups - we did not consider a sentient object type solution and did not focus on dependability. However, an approach based on sentient object behaviour would appear to be a possible solution for achieving decentralised control - especially given the need to maintain levels of dependability in this essentially safety critical application domain. Such as approach could utilise agents, e.g., safety agents, in order to augment the distribution of control and support cooperation.

3.3.1 Autonomy

The following list provides a sample of the type of autonomous agents that could be used in this application scenario:

- safety agent - performs autonomous network switching in order to ensure the safety of an engineer.
- task agent - manages the tasks of a human field engineer, e.g. when a given network operation should be carried out.
- scheduling agent- allocated resources required to perform a schedule of work - this is likely to be achieved in a context-aware manner - for example which engineers are geographically close to the fault area, etc.
- control agents - take the work of augmenting the role of a distributed control centre - these define zones of dependability.
- awareness agent provides a field-engineer with an awareness of activity of others.

3.3.2 Consistency

We envisage the notion of 'Zones of consistency'. Agents need to work on some level of consistent network view - within zone - where zone relates to a zone of dependability. This zone can therefore be dynamic based on parameters such as connectivity.

There is clearly a strong requirement for consistency within a zone and possibly overlap between them. Timeliness is also important with respect to maintaining consistent views. For example, consider 3 engineers, A, B, C such that at some time they all have consistent global view. Now A receives an update to the global view that is not received by B and C at the same time. The result is that partitioning takes place such that A is effectively operating within one zone of consistency while B and C operate within another.

Timeliness of propagation of state change is (context) dependent upon the criticality of the update, e.g. a change in location of a field engineer is probably less critical than notification that a given HV feed has become live.

Another dimension that pervades this environment is trust. Trust is a difficult and unresolved issue in fixed environments such as the Internet, but is even more difficult when you introduce elements of sentience, mobility, disconnection, autonomous and anonymous execution, etc. In safety critical situations it is however crucial to be able to trust available

data, advice from agents, etc. it is therefore a very interesting issue to investigate how elements of trust (e.g. of consistent views) can be provided in such sentient environments.

3.3.3 Cooperation:

The need for co-operation manifests itself in a number of ways

- Cooperation between safety agents negotiating fail-safe behaviour.
- Cooperation between agents and field engineers and control centre though control centre may not be decentralised.
- Cooperation between control agents and safety agents
- Task agents cooperate with awareness agents.

3.3.4 Requirements Analysis

3.3.4.1 Requirements from the environment

UIS does address:

- Large number of physical entities: e.g. switches, cables & other active network components. Teams of engineers (simulated/ emulated).
- High geographical dispersion of entities: Entities (see above) would typically be dispersed over at least a countywide area (optionally country wide).
- Human safety relies on computer-controlled entities: In a 'real' deployment situation, the application is highly safety critical – both in terms of Engineer safety and power supply to critical/ high priority users (e.g. hospitals, dialysis machines etc.).
- Computer control of entities requires timely response: in order to make sure that changes to the physical network state are consistent and are propagated in a timely manner to ensure that coordinated behaviour is achieved.
- Hierarchies of dependability/ zones of consistency: Power distribution networks have implicit hierarchy (high voltage, low voltage etc.) and task federation (e.g. delegation of repair tasks which occur within a specific localised region of the network).

UIS does not address the following:

- Subset of entities should be highly mobile: the network entities are stationary and field engineers change location infrequently (with respect to changes in network state).
- State of entities does not change rapidly: network state changes occur only after careful and coordinated action. However, once a change occurs state propagation must occur with strong real-time constraints (i.e. speed of light!).
- Feasible to demonstrate: the implications of real world deployment, which is not viable given the physical and financial constraints of the Utilities Industry.

3.3.4.2 Requirements from the computer system

UIS does cover:

- Support for scalability: Potential number of entities that must be modelled/ simulated is large (digital representation of the established physical infrastructure is an ongoing issue due to the complexity of the system).
- Support for real time computing enforcing hard QoS requirements: there are strong real-time requirements (see above).

- Support for fault-tolerance: the system should react in a consistent and well-defined way to cope with to partitions and failures.
- Support for autonomy: network entities, safety agents etc. must be capable of autonomously behaviour.
- Support for sentience: network entities and software agents guarding personal safety must have limited sentience (e.g. detection of a cable fault, triggering action in a higher level safety agent).

UIS does not cover:

- Support for adaptability/evolution of the system: the system only evolves in terms of interaction between the various agents & entity state.
- Support for heterogeneity and openness: a real end-user system would need to address these issues (due to the federation of the power supply function within UK). However, within our prototype this is not a clear focus.
- Support for mobility with wireless communication: the simulation / emulation environment need not require mobile computation nor wireless communications (these may be simulated). Field prototypes may have such requirements.

3.3.4.3 Techniques/mechanisms to be used

Useful in UIS:

- Real-time networks: Timely propagation of network state changes (or actions that affect such state) is a high priority.
- Cooperation mechanisms: Sharing of consistent views of network state is an essential part of the system.
- Group communication: Group semantics and QoS specifications on group communications are important for ensuring management of timely state sharing, network partitioning etc.
- Event-based communication.
- Filtering: filtering in terms of grouping, proximity to the distribution network or by distribution hierarchy, may be important in UIS.

Not useful in UIS:

- Actuator technology: Real network switching actions would be possible with actuators (not required for simulation or emulation however).
- Sensor technology: Power distribution sensors (cable liveness, switch state etc.) would be needed in a real deployment scenario.
- Anonymous communication.
- Wireless networks.

3.3.5 Feasibility

Due to the highly safety critical nature of the Utilities Industry, full end-user participation is unlikely. However, we have experience of developing fault scenarios within this domain and believe that we have sufficient expertise in order to develop tailored fault scenarios in order to exercise the criteria outlined above. Furthermore, we envisage developing a simulation/ emulation environment in which to construct working prototypes to exercise specific fault scenarios. Such scenarios do not place strong requirements on specific hardware and software platforms.

3.4 Assisted Terrestrial Transportation System Scenario

In modern societies the role of "time" is of the utmost importance. There is continuous pressure to perform actions according to predefined schedules or within given time intervals. Typical examples are: waking up at 7h30, arriving at work before 8h30, lunch in 1 hour, go to a meeting on the other side of the town at 15h00, pick up the children at school before 18h30, etc. However, due to the inherent scale, openness and complexity of the environment in which we live, there are many non-controllable factors that introduce uncertainty on the time it takes to accomplish certain things, such as traversing a town during rush hour.

The vision of the CORTEX project is that modern technologies can be used, with the help of adequate architectures, to address issues like the one described above. The scenario described below, while putting in evidence these timeliness aspects and the uncertainty introduced by the environment, allows the establishment of requirements for the CORTEX infrastructure. This scenario is centered on an Assisted Terrestrial Transportation System (ATTS), whose objective is to help individuals to arrive at a certain destination before a given deadline, or within a certain amount of time.

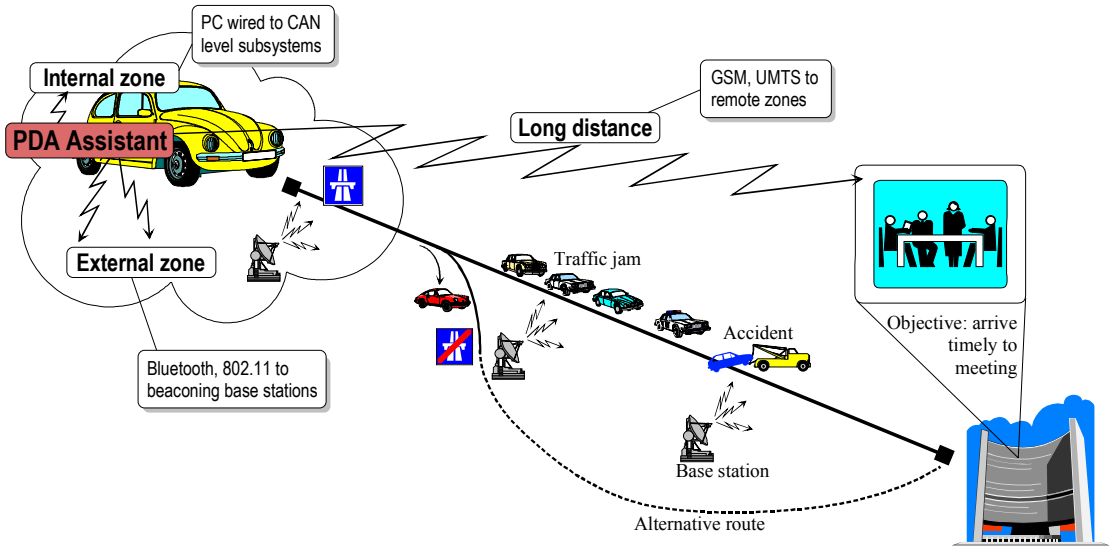


Figure 3.1 ATTS scenario

Figure 3.1 highlights the essential aspects of this scenario. The goal of some car driver is to reach a distant location to attend a meeting that will take place at a given hour. There is at least one highway connecting the starting point to the arrival point. It is possible that several other routes also connect the two points. The ATTS would typically be deployed on a PDA assistant, carried by the car driver or provided as an integrated car instrument.

The essential function of the ATTS is to indicate the route that minimizes the time it takes to arrive at a certain place. However, it is obviously impossible to know a priori all the facts that might influence this calculation. Therefore, the idea is to have an ATTS that provides the probability associated to each calculation, that is, the probability of arriving within a certain amount of time to a destination taking a particular route. The probabilistic aspect is key to the problem. For example, if there are two routes that take us to some place, a faster and a slower route, one might opt for the slower one if the probability of arriving before the specified time using this route is higher than for the faster route. Note that each route is exposed to different factors, which may introduce different degrees of uncertainty. The user must specify the risk that one is willing to take.

The probabilistic aspect has another facet, which is related with dynamic factors. The changes in the environment that occur during the course of the trip may cause significant disturbances

on the time it takes to arrive at the destination. For instance, an accident in the path to the destination may introduce an extra delay, or an additional uncertainty, or both. The ATTS handles this kind of event by re-evaluating the risk incurred in taking the current affected route or possibly changing to some alternative route.

To be able to do its job, the ATTS needs a large amount of information. There are two basic classes of information: static and dynamic. Static information includes the distance to the destination, the maximum allowed speed of the roads and the vehicle type. Dynamic information includes the weather, the traffic intensity and the condition of the car. This information can be obtained from three different sources: from the internal zone (the vehicle), from the zone surrounding the vehicle, or from distant zones. From the internal zone it is possible to obtain information related to the vehicle, like the available fuel or the fitness of the breaks. The external zone delivers information such as traffic jams, obstacles in the road, nearby gas stations or highway exits. Finally, other kinds of information like the weather forecast or changes to the scheduled meeting time can be obtained from remote zones.

The infrastructure to support all the above information flows makes use of several existing technologies, and fully exploits the WAN-of-CANs structure envisaged in CORTEX. The CAN level is composed by the several micro-controlled subsystems of the car, which operate under very strict timeliness and dependability requirements. The aggregate of these subsystems constitutes the internal zone, which can be accessed through a gateway device, for instance an embedded PC. The assistant PDA may connect to this PC in order to obtain information from the internal zone. Eventually, each car is equipped with similar subsystems and has a gateway device. From an external point of view, each car can be treated as an independent and autonomous object, capable of generating events or information useful in its surrounding environment, and capable to react to events disseminated by others.

The communication among all these CAN islands (cars) is accomplished by means of events disseminated to and received from external zones. Beaconing base stations spread along the full length of highways have the capability of detecting the presence of cars (and the events they generate) passing through the zones that they are serving. Furthermore, these base stations may possibly be interconnected by a backbone. The event communication can be implemented using one of the existing wireless technologies for (relatively) short distances (e.g. Bluetooth or the 802.11 standard). For the backbone, a solution can be the use of fiber optical cables, or other infrastructure that is already in place (e.g., using power lines for data transmission). The combination of the cars and base stations obviously forms a WAN-of-CANs structure.

Connecting to remote zones can be accomplished by the use of technologies such as GSM or UMTS. In this case, the distance is not a limiting factor for the communication. But the QoS that can be delivered by this communication infrastructure is possibly lower that the QoS provided in external zones, and certainly much lower that the QoS required in internal zones. This has obvious impacts on the kind of information that can be exchanged over long distance links. Their purpose is fundamentally to allow sporadic access to remote data or information servers.

At least two communication paradigms can be employed in this assisted terrestrial transportation scenario. The publish/subscriber paradigm is present in the communication to external zones. Events are published to the environment (captured by the beaconing base stations), eventually propagated to other zones, and captured by other objects (cars) that have subscribed to those events. The same paradigm is also employed in the communication in internal zones: each subsystem inside the car publishes its own events and subscribes to the events published by others.

The client/server communication paradigm is employed in the communication to remote zones. Long distance communication typically serves to obtain specific information, upon request, from particular servers. Cars act as clients and remote systems act as servers.

3.4.1 Autonomy

Autonomy is clearly present in this scenario, with each of the vehicles moving on its own, independently of the others, in a completely autonomous manner.

3.4.2 Cooperation

Requirements for cooperation and coordination are minimal in this scenario. They can be derived from general safety requirements, such as avoiding accidents, or from particular requirements related with improving the functionality of the ATTS. In the former case, cooperation can be useful to propagate (warning) events such as those indicating the existence of obstacles in the road, or slow vehicles in front, or fast vehicles (e.g. ambulances) behind. In the later, the ATTS can be improved if there is some coordination among all ATTS instances when decisions have to be taken. For instance, if the traffic is slow due to an accident, the decision as to whether to take an alternative route or continue on the same route should be coordinated. Only a fraction of the vehicles should take the alternative route. Such a coordinated action would provide some kind of optimized traffic management and more accurate predictions of the ATTS.

3.4.3 Consistency

Consistency is not a particularly important attribute in this scenario. It is obviously convenient that every object (car) has a consistent view of the environment (what is happening in the highway), but it is not strictly necessary. Otherwise, the requirements for cooperation would certainly be much higher, and autonomy would possibly be reduced. The lack of a consistent view is not critical. If a car misses some events and does not get to know about a traffic jam, at worse it will be delayed and will arrive late to its destination, while some other cars may take a different route to avoid the queue.

Although in this ATTS example there are a considerable number of requirements imposed on the supporting infrastructure, they do not include the aspects of predictability or guaranteed QoS. In fact, the operation of the ATTS only imposes soft timeliness constraints and remains correct despite the occurrence of sporadic timing failures. Unpredictable behaviors such as those resulting from shortages of the available communication bandwidth, from variable communication delays or from sporadic losses of connectivity, do not necessarily compromise the correctness of the system. In other words, fluctuations in the QoS delivered by the supporting infrastructure can be easily tolerated if adequate measures are taken. Nevertheless, since the system is not totally best effort, there are still some problems resulting from the variable end-to-end QoS that must be addressed in CORTEX. For instance, it is necessary to prevent old information (carried by events that take too long to be received) from being used when computing some decision.

A possible classification of the ATTS scenario could be derived from the type of problem that is being addressed, which in this case is the problem of *adaptive optimization of goal achievement*. To a certain extent, and using the same classification criteria, it is possible to include the ATC scenario in the same category.

3.5 Remote Control of Real-time Operations Scenario

One of the areas where the use of real-time systems and real-time architectures is of substantial importance is the area of real-time control and real-time embedded applications. There exist centralized and distributed control architectures. Independently of the approach that is used, the requirements on predictability impose hard constraints on the infrastructure, and implicitly limit the distance between the controlling and the controlled subsystems. Therefore, real-time control is usually not considered in geographically large-scale settings. As a matter of fact, it is usually granted that long distance control can only be done on a best-effort basis.

We propose a scenario that deals with a more ambitious goal, which consists in the remote control of real-time operations. While it may be unrealistic to consider real-time properties for the overall control loop, we suggest that more than best-effort control may be achievable. The scenario describes an application that introduces these increased predictability requirements.

Figure 3.2 illustrates the proposed scenario, where the idea is to have an application that is able to remotely control a small aircraft (other real-time subsystems could be used, but this is sufficient for the sake of our purposes). There exist two levels of control. Real-time control is used inside the airplane to ensure that essential operational functions are executed in a timely manner. Non-functional control is used to remotely control the behavior of the aircraft. The former is achieved on top of some Controller Area Network (CAN), which exhibits the required synchrony properties. This real-time communication environment forms what we call an *internal zone*. The latter is done using an infrastructure adequate for *long distance* communication.

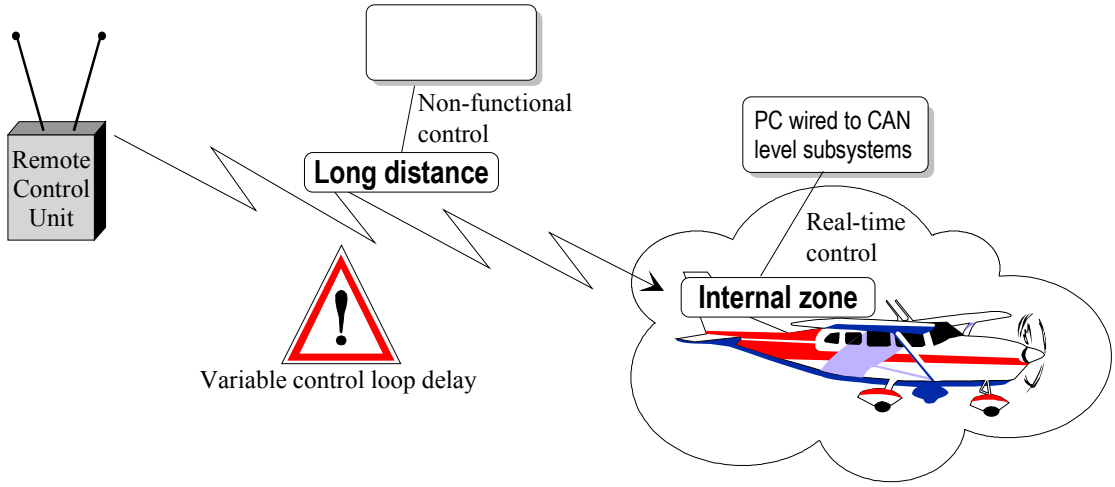


Figure 3.2 Remote control of real-time operations scenario

The major challenge to be addressed in this scenario derives from the variable control loop delay associated with the long distance communication. It is necessary to handle this unpredictability in such a manner that some useful properties can still be exploited to allow non-functional control to be achievable.

In terms of infrastructure requirements, this scenario suggests the need for two basic communication environments. One to support the real-time control activities performed in the internal zone, and another to support the remote non-functional control. In the former case, an obvious solution is to have a PC (the control unit) wired to CAN level subsystems by means of some fieldbus network (e.g. CAN). For the latter, existent wireless technologies such as GSM or UMTS may be envisaged.

An interesting aspect of this application scenario is that it provides the opportunity to employ different communication paradigms. For example, remote control of the real-time operations can be performed using remote procedure calls (RPC), following a client/server model where the control unit acts as a client and the controlled device acts as a server. On the other hand, real-time control in the internal zone may be based on dissemination schemes, with messages transmitted during specific time slots. To some extent, this form of communications follows a publisher/subscriber model.

3.5.1 Autonomy

Some of the main attributes of the applications that CORTEX pretends to address are present in this scenario. For instance, the aircraft can be considered as an autonomous object. Its autonomy is granted within some bounds, since the low-level operational decisions are taken without any control of the remote control unit. The autonomy is limited by the remote control actions, or orders. Geographical dispersion derives directly from the long distance control of the real-time unit, and time-criticality emerges from the need to perform real-time actions in the internal zone. The application may also be safety-critical, if one assumes that the aircraft make take passengers on it. Finally, the problem of mobility is obviously present, and the application also has to cope with the evolution in the environment conditions that affect, among other things, the control loop delay of the non-functional control.

3.5.2 Consistency and Cooperation

In the scenario described so far the need for cooperation and coordination is clearly absent. Equally, attributes like sentience and large scale are practically not present. Because of this, we suggest an extension to the scenario in order to increase the complexity of the application and consequently the problems that must be addressed in CORTEX. The extension consists in assuming that instead of a single aircraft there exists a fleet of flying planes, of which only one (the fleet leader) is remotely controlled (see Figure 3.3).

In consequence, several important requirements are raised. Consistency is needed to ensure that every plane has the same view (in time and space) of the environment, so that they can know each other's positions and crashes are avoided. Coordination is needed to make specific formations during the flight, or particular flight sequences. Cooperation is clearly necessary to achieve the previous goals, and also to ensure safety properties. For instance, it is needed to ensure that every plane follows its leader, and maintains a certain distance to planes flying side by side.

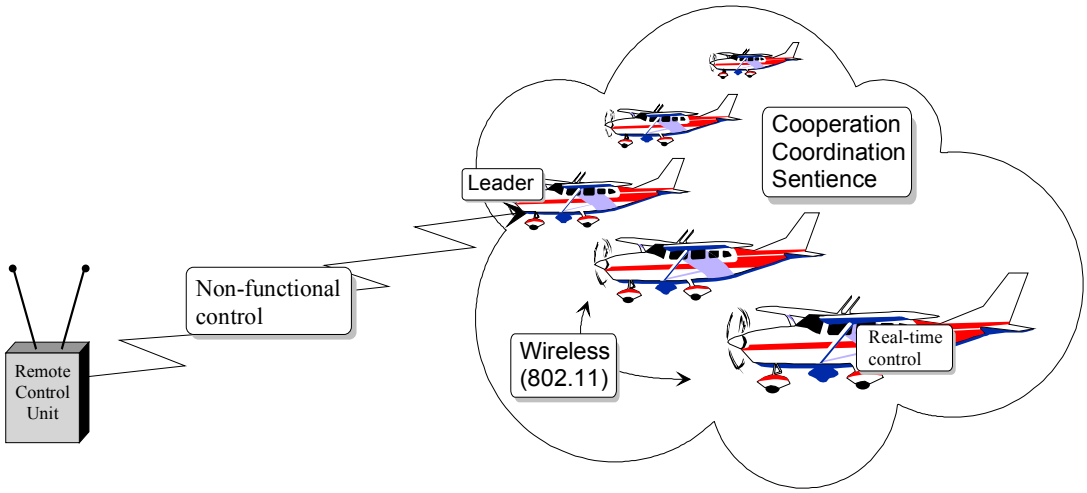


Figure 3.3 Extended remote control scenario

This extension also has implications from the point of view of the infrastructure. A short-range wireless communication infrastructure (e.g. 802.11 standard) is now needed to support the communication between the planes. The set of properties that should be enjoyed by this interaction among planes is another aspect to be dealt with in CORTEX. For example, there obviously exist some timeliness requirements that must be verified in order to achieve some degree of time coordination.

In general, the scenario just presented can be classified in the class of those that address the problem of *remote control of semi-autonomous (cooperative) entities*. While the problem of dealing with the variable control loop delay is perhaps where CORTEX could be focused, the

application proposed in this scenario can be considered quite complete from the point of view of the aspects that CORTEX aims to address. In fact, although none of the fundamental autonomy, cooperation and consistency attributes is exacerbated, it is possible to say that in this application they are all equally and significantly required.

3.6 Teleoperation Scenario

Teleoperation of (semi) autonomous sentient systems become an important issue in the areas of exploration of unknown dangerous terrain, mine detection and deactivation, search and rescue operations, remote mining applications, telemedicine and even house appliances. The essence of theses applications is to remotely control an actuator, an electronic device or a mobile entity with a human operator in the loop. The operator will not blindly perform the control but will be assisted by the system to perform the control task (this extends the Remote Control scenario described in section 3.5 that implicitly requires visual contact with the controlled object). The problem in controlling the remote device by the operator arises from long and probably unpredictable delay and jitter of the messages between the operator and the actuator. Today, remote control relies on the availability of an infrastructure that guarantees the temporal and functional needs of the control channel within very tight bounds. This substantially raises costs, promotes proprietary special solutions and is geographically restricted to rather small areas. However, if available infrastructure such as the Internet is used to bridge large distances and standard and cheap wireless ad-hoc and multi-hop networks are part of the communication channel, it will be impossible to make strong assumptions about the predictability of the communication link. Moreover, inherent communication delay over long distances or slow media will prevent an immediate response of the remote actuator and feedback to the operator. Figure 3.4 shows the components of such a system on a technical level.

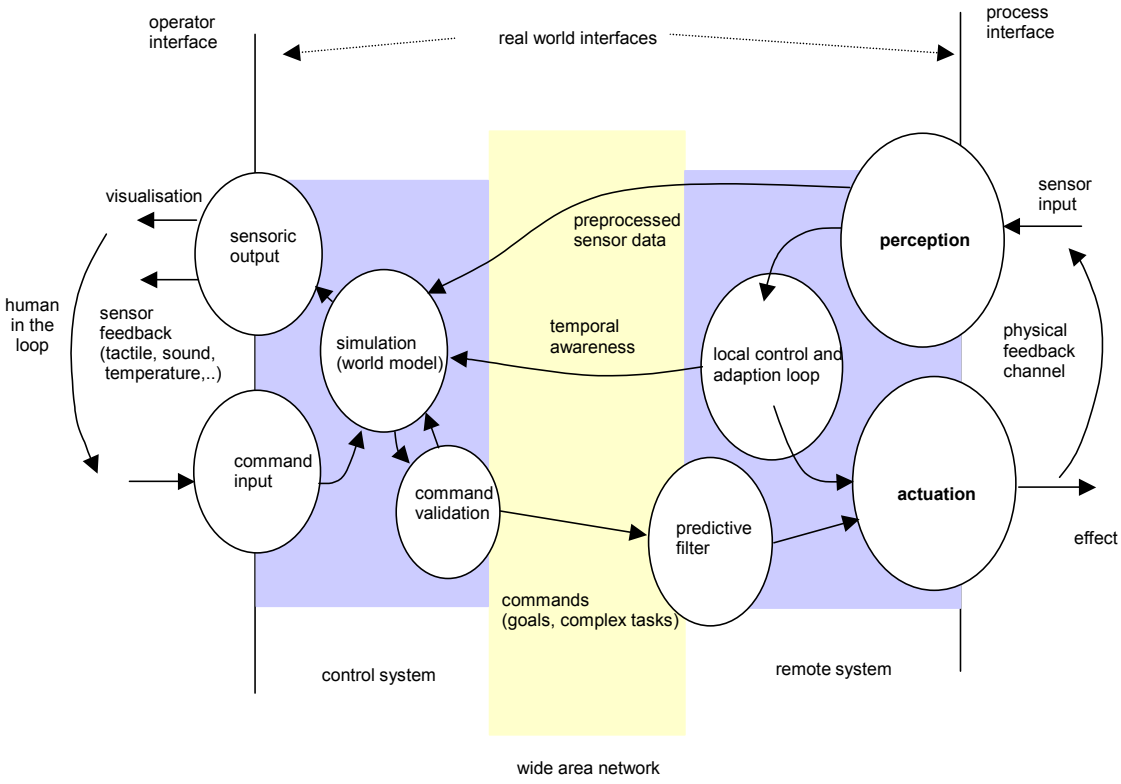


Figure 3.4 General structure of a remote control application scenario

To control the remote component, the operator needs adequate input devices ranging from a simple keyboard, graphical pads, or a mouse to data gloves and special application-specific components. Because the effect of the control input cannot be observed directly, visualization of the actuator, e.g. a camera image or tactile (force) feedback has to be provided. The control input may be fed to a simulated model of the actuator. This has two purposes: firstly, the control input may be validated that it does not violate any physical constraints of the actuator (e.g. validation of the movement of the joystick against the flight characteristics of an airplane). Secondly, the feedback to the human operator may be required immediately in an estimated way and later be adapted by the real feedback coming from the sensor input of the remote actuator.

Because of the delay and unreliability of the communication connection, which may be substantial, the remote actuator needs a certain degree of autonomy, i.e. it must be able to act in the presence of a temporary network failure or unexpected delays. In some cases, it may be desirable to transmit only goals (e.g., a robot should move to a certain place, or perform a complex tasks).

The control component and the remote component may constitute islands of tight control, which incorporate multiple smart sensor and actuator devices and a local feedback control loop. Therefore, in these systems, the message delay and jitter can be bound. For the communication network, tightness and precision may be very large. Although, even if it would be possible to derive a bound on network delay, this may be impractical to use in a worst-case scenario as the usable bandwidth would drop dramatically.

Figure 3.5 depicts an application that remotely controls a grabbing device.

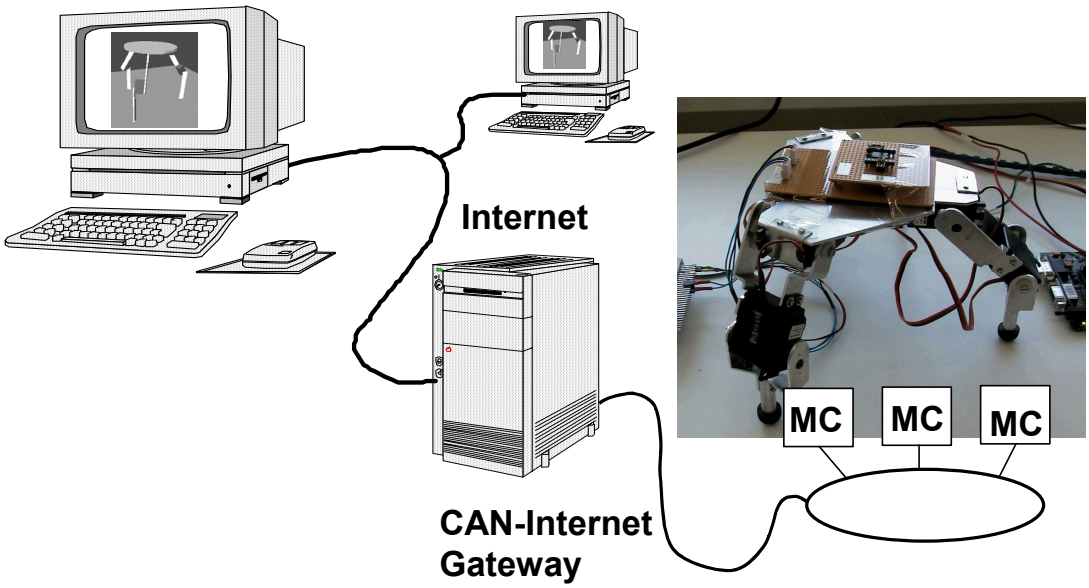


Figure 3.5 Remote control of a grabber

The grabber may be controlled from different places, one at a time. The screen of a control system, as part of the operator interface shows a visualisation of the grabber and provides some means to provide input commands. The remote real device comprises the actuators and feedback sensors to control the local feedback loops and provide sensor feedback to the operator. The grabber itself is a distributed sensor/actuator system connecting the various smart components by a local CAN. The same may be true for the operator interface which transfers the received sensor information e.g. into force and temperature for a haptic/tactile user interface. Additionally, e.g. inclination sensors, angle detectors, optical, ultrasound or infrared range sensors e.g. at the fingertips will be used to perceive the orientation of the

grabber, the environment and the position of the object which has to be grabbed in a three-dimensional space.

There may be more than one grabber involved in a more complex task of detecting and salvaging an object e.g. in an exploration, a search and rescue, or a remote repair application. This would require a tight coordination of activities. In this case while controlled by multiple human operators the remote components would also directly interact to achieve the necessary level of reactivity in a safety critical situation, thus forming a complex distributed control system. Consequently, we observe a three level hierarchy of networks with decreasing demands on the predictability. Firstly the local wired CAN inside an autonomous grabbing device with the highest level of predictability (this can be refined recursively to even lower levels of the system as autonomous subparts of a grabber). Secondly, the local wireless interaction between autonomous grabbers. This interaction is also substantially controlled by the physical feedback channels (see Figure 3.4) that also raise mutual awareness between the acting components. Thirdly, the Wide Area Network (WAN) that has a low level of predictability. However, it is very important for the planning of local safety critical actions that there is a high degree of awareness about the functional and temporal status of the WAN. On the basis of this information, local actions have to be autonomously adapted. This is a major point of research in CORTEX.

3.6.1 Autonomy

In the above example of the fine grain control of a mobile grabber, autonomy is crucial to achieve an acceptable degree of safety because of unpredictable characteristics of the WAN. Information on the temporal and functional status of the WAN are needed to adapt autonomous behaviour (WAN monitoring). However the degree of autonomy may differ widely. There may be very high-level commands like: explore area (coordinate x, coordinate y) and alert the operator to a specified event. In this case the actuator, e.g., a robot, is completely autonomous.

3.6.2 Consistency

In this scenario, consistency requirements are stringent. It should be noted that in the basic scenario of tight remote control, only few entities must have a consistent view. In the case where multiple operators commonly manipulate multiple actuators, consistency is strongly required, locally between the (semi) autonomous actuators as well as between the distant human operators. Consistency constitutes a safety critical property in this scenario.

3.6.3 Cooperation

As described above, cooperation may take place on different levels. Because the scenario includes safety critical aspects, coordination must be achieved in a predictable way in the functional and temporal domain. This has to be achieved over the potentially unreliable WANs and hence the scenario aims at one of the main objectives in CORTEX.

3.6.4 Requirements Analysis

3.6.4.1 Requirements from the environment

- Sentience is obviously one of the crucial properties of this scenario. All kinds of sensors are needed to provide the necessary autonomy of local actuator operation as well as providing the necessary feedback for the human in the loop.
- Scale: Moderate. In the grabber example it may be rather low. In the case of exploration of terrains it could be thought of a few hundreds of small robots coordinating themselves. The platoon of planes is another example of a larger scale application.

- Safety Critical: Potentially.
- Geographical Dispersion: The geographical dispersion of the application is low. However, human control and machine actuation are assumed to be distant.
- Mobility: Potentially.
- Evolution of environment: Actuation will change the situation dynamically (as described in the CORTEX proposal). The physical feed back loop at the remote site is an important property of the application scenario. However, from the point of system evolution, it is a less demanding scenario.

3.6.5 Feasibility

In general, the basic scenario appears to be feasible within the CORTEX context. The degree of autonomy of the controlled entity may differ widely. There may be very high-level commands like: explore area (coordinate x, coordinate y) and alert the operator on a specified event. In this case the actuator, e.g., a robot, is completely autonomous. This is also related to aspects of the intelligent in-house delivery scenario (section 3.8). Also, relations to the power-switching example (section 3.3) can be seen in which an operator may control a large number of (smart) switches.

3.7 Cooperating Cars and Floating Car Data

Car-to-car communication has three major goals:

1. Dissemination of traffic information derived from the cars' embedded sensors.
2. Cooperation of cars to assist the driver in critical situations.
3. Interaction between remote cars.

We will discuss these scenarios and derive the specific needs for the communication system.

Dissemination of traffic information derived from the cars' embedded sensors.

The car of the future will be equipped with a large number of sensors, ranging from position and speed sensors to sensors indicating road and weather conditions and sensors signalling braking events. All these sensors will be accessible by the internal control network (CAN) of the car. To enable exploitation of the sensor information by other cars, it will be disseminated via a wireless link to a larger network, thus forming a WAN-of-CAN structure as envisaged in CORTEX. Therefore, it will be possible to derive certain traffic conditions directly from the sensor information provided by the floating cars themselves rather than by a fixed roadside infrastructure. The respective information is referred as floating car data.

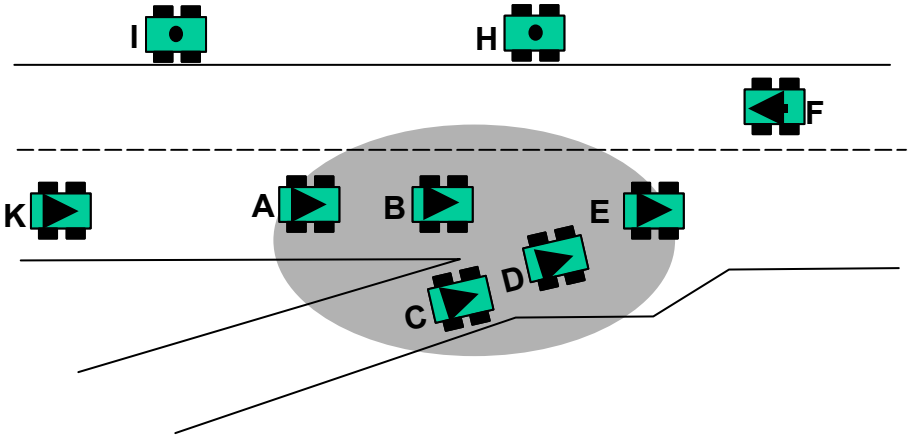


Figure 3.6 Traffic scenario

The principle characteristic of the sensor information is its aging in time and that it may only have a geographically limited relevance. When other cars receive such information, they must be able to assess the relevance of the information according to the temporal and spatial parameters. Moreover, the communication system itself may exploit these characteristics when disseminating the information and consider the context in which the information was generated as a basis for constraining the dissemination over the network in order to reduce the overall network load. We may define an area (or zone) of relevance for the information in the temporal and the spatial domain in which each car has access to the information.

An important aspect is the correlation of low-level sensor data to infer some higher-level events. For example, a traffic jam may be concluded from a large number of speed sensors, which indicate very slow moving cars. Bad weather conditions from a large number of rain sensors or the fact that many cars switched on their windshield wipers. The combination of low-level sensors to a sensor detecting more complex events is denoted as sensor fusion, the resulting sensor is called a virtual sensor. It should be noted that the correlation of multiple sensors has to be performed in the receivers, which requires the general availability of sensor information. The number of sensors is very high, and, due to mobility, varies dynamically. Therefore, a point-to-point, client/server model of communication to disseminate the information is not appropriate for this purpose. Rather, an event driven, generative communication model will be necessary. Techniques of information diffusion and a model of a shared information space in which an object may take the role as a producer or a consumer of information will be applied. In summary, we can observe the following properties:

- The information of deeply embedded sensors is correlated and exploited to derive some useful traffic information.
- The information is used to raise awareness of the affected cars. The relevance of the information decreases with the geographical distance to the detected conditions. Although this is true in general, it should be noted that even distant conditions could be used by a car that may be far away to re-plan the route it has to go (as in the ATTS scenario).
- The information may become critical when the car approaches a certain proximity to the event detected. (see point 2. below).

The car-to-car communication medium will be a short-range wireless link that enables direct communication within some hundred meters in free terrain and less in urban environments. This is motivated firstly by the nature of data (described above) and by costs. The car manufacturers want to exploit a license free communication medium for their customers. Therefore, a multi-hop network will be established exploiting other cars as relays. Because the cars will move, it is possible to transport information over a long distance even if there is

temporarily no link to disseminate useful information to interested groups of cars directly. In Figure 3.6 car F will provide information concerning traffic conditions ahead for the other cars. Cars I and H may relay this information although they are parking at the roadside.

Cooperation of cars to assist the driver in critical situations

In order to prevent critical situations and accidents on the road, to improve throughput and to increase the comfort for the drive car-to-car cooperation is an important goal for future traffic control systems. The car control system is always aware of its specific geographical and traffic context. Consider the situation sketched in Figure 3.6 with cars A, B, E moving on a main road. At the junction cars C and D want to join the main road. At that point, communication is needed to coordinate the speed of the cars to smoothly merge the traffic. It is clear that if the responsibilities of actions are moved from the driver to a control system, communication has to achieve a high degree of predictability. The procedure requires the information derived from the sensor systems of the cars is communicated, an optimal speed is negotiated between cars B and D and the control system takes the respective actions. Subsequent cars A and C may be involved in the whole procedure and have to adjust their speed accordingly. The more distant car K may only be aware of the situation in detecting a speed reduction ahead by means of the disseminated sensor data.

Interaction between remote cars

The third goal is to allow the interaction between cars that are geographically distant. Therefore, direct communication is not possible. Examples are the coordination of platoons of cars, coordinating the handover of goods in a transportation chain or just connecting individuals travelling within the car who want to communicate. The difference to point 1 is that here a well-defined group of cars needs to communicate and coordinate actions rather than just disseminating traffic information. While disseminating traffic information is an anonymous best effort task, interaction between cars needs a higher degree of predictability. The difference to point 2 is that the coordination is not safety critical. A temporary loss of the connection should be prevented but may not have safety-relevant consequences. The main challenge here is to predict the connectivity of the interacting cars from the range of a direct connection and the sensor information like position, speed and intended direction of cars that relay the messages. Thus, the floating car data form the basis to maintain the link between cars. A loss of connectivity can be predicted before it actual happens. Then, dependent on the importance of the link, communication may be switched to a long distance point-to-point medium like a GSM connection.

3.7.1 Autonomy

Certainly, cars are autonomous entities. In case of cooperation, control autonomy is not affected. All control decisions are made on the basis of local sensors and the negotiated parameters.

3.7.2 Consistency

In case of cooperation, the cooperating entities need a consistent and agreed view. This is highlighted by the examples given above.

3.7.3 Cooperation

There are two kinds of cooperation possible. One is the direct cooperation in safety-critical situations like joining the traffic flow at a junction as described above. Another form of cooperation is a loose cooperation supported by a best effort communication over a multi-hop network. This form has relations to the platoon of planes in the remote control scenario.

3.7.4 Requirements Analysis

3.7.4.1 Requirements from the environment

- Sentience: Due to the embedded sensors, cars provide a rich context. Already today, modern high-end cars have built in around 60 processors, a local network (CAN) and sensors that cover almost any driving situation and road condition. Brake sensors may signal a critical situation, from the active suspension control, the road conditions can be inferred; speed and position sensors may indicate a congestion or a traffic jam. An important feature is the fusion of multiple physical sensors to a virtual sensor that may be used to detect completely new events. The data generated by the sensors are termed floating car data and form a new opportunity to regulate the traffic in a completely decentralized manner on the basis of a car-to-car communication facility.
- Large Scale: The scale of the scenario can be considered as large. Thousands of cars may be involved with some ten thousand embedded processors and sensors.
- Safety Critical: Obviously, traffic scenarios are inherently safety critical. Particularly, in the tight coordination tasks that are described in the scenario, an ultimate level of predictability and safety are necessary. To turn the vision of automatic traffic regulation to reality, safety will be the decisive property.
- Geographical Dispersion: There are different degrees of geographical dispersion with different requirements. Cooperation takes place in a geographically confined area but with high demands on predictability. On the other hand, traffic information derived from floating car data, may include a wide geographical range. However, because typically only awareness about certain conditions on the road ahead is provided, the requirements about the temporal and functional properties when communicating and processing these data is are much lower.
- Mobility: Mobility is an intrinsic property of the scenario.
- Evolution of environment: The environment changes rapidly. Decisions based on the actual traffic situation will change the situation dynamically forming a feedback loop as described in the CORTEX proposal. There are cars from many manufacturers with a widely differing set of properties that develop with each generation. Therefore the ability for system evolution is a basic design requirement.

3.7.5 Feasibility

Aspects of the scenario can be realized with cooperating autonomous robots. However, experimentation with real cars can only be done in cooperation with a car manufacturer.

3.8 Smart (in-house) Delivery System Scenario

Smart in-house delivery systems generally comprise the following cooperating subsystems:

- smart vehicles which carry a payload;
- smart payload;
- a smart infrastructure (environment).

A smart in-house delivery system conveys all kinds of payloads via autonomous intelligent vehicles. A smart delivery system may range from a simple transportation system that transports an item with one vehicle from one place to another to a complex transportation system in which cooperating vehicles form an optimal delivery chain. In the latter case, vehicles will advertise spare capacity if available as well as their routes. They will also be able to look-up advertised routes. By calculating distances, estimating speed etc., vehicles can dynamically negotiate contracts with other vehicles to bring an item to a certain destination. The task is supported by smart payloads that provide information about the destination, the

sender, urgency and deadline. The payload itself may determine the kind of transportation required. Many options are possible, e.g., an expensive exclusive vehicle which brings it from A to B (taxi). An optimised route may be shared by other payload items (shuttle). A standard delivery route may stops at many places and can be used if free capacity is available (bus). A smart environment supports global communication, provides discovery and lookup services and performs localisation and navigation tasks. The environment also has sensors and actuators like automatic doors and elevators that cooperate with the mobile entities. It may provide local maps, local directories and information about available resources.

There is a strong relation to all scenarios that use autonomous robots because the transportation vehicles constitute autonomous components that sense the environment and autonomously negotiate and take decisions. Similarly, they provide floating data from the sensors like speed, position, and free capacity as well as higher-level information like intended route, costs, etc. This information is used when trying to find an optimal cost function for the transportation. Thus, the scenario also exhibits the WAN-of-CAN structure, connecting the internal system of the vehicle to a larger and less predictive network. In this respect, the scenario has a relation to ATTS that also tries to find an optimal route to meet high-level timing and deadline requirements.

There are many situations in which vehicles have to operate under safety critical requirements. Obviously, like any autonomous vehicle, obstacle avoidance and emergency stop demand such properties. Critical operations are also exchanging payload with other vehicles or coordinating speed at intersections. In these situations, the problem is related to the one discussed in the cooperating cars scenario. Additionally, the vehicles have to cooperate with the infrastructure, open doors and use elevators. This may also require a high predictability of operation.

The environment for a vehicle comprises static elements and dynamic elements. Static elements are represented in maps. It is assumed that maps exist ubiquitously and can be requested from the smart environment. These static elements include doors, rooms, elevators, etc. The dynamic elements can be seen in two groups, a group that provides elements with an auto-locate property, i.e. these elements move, but their position can be indicated as coordinates in the map. The other group is entities without these properties. This group comprises moving elements like humans, animals or non-moving items that are temporarily put down on the floor. This group will be abstracted as obstacles. If many obstacles are on the way, the speed will be reduced which in turn will be detected by other vehicles that may change the cost function for this route.

3.8.1 Autonomy

Vehicles form completely autonomous entities.

3.8.2 Consistency

A consistent view is desirable to plan a route efficiently. However, strict consistency is only required between contracting partners.

3.8.3 Cooperation

As described.

3.8.4 Requirements Analysis

3.8.4.1 Requirements from the environment

- Sentience: Sentience and environmental awareness are crucial for the autonomous vehicles. Environment awareness may be based on two basic mechanisms:

1. As described above, the static properties of the environment can be derived from maps that can be downloaded dynamically from available local services. This must be complemented by the possibility of the autonomous vehicle to determine its actual position.
2. The dynamic properties of the environment, subsumed as obstacles have to be sensed by the vehicle as it moves. For this purpose, a plethora of distance and scanning sensors are available.

- Large Scale: In large buildings this can be some hundreds or thousands of smart components. Particularly, in addition to the vehicles, the scenario includes all the sensors and the services of a smart environment and the smart payloads.
- Safety Critical: As in all scenarios dealing with mobile autonomous vehicles, safety is a major concern. Mechanisms have to be provided to maintain a safe behavior of the vehicles under all circumstances. This includes safety properties of the local system when internal failures occur and a safe behavior with respect to the environment. The first property can be derived from the broad body of research in fault-tolerant systems. The second property relates to environment sensing and reliable cooperation protocols.
- Geographical Dispersion: The geographical dispersion of the basic scenario should be confined to a building. However, it could easily be extended to a global scale if goods have to be delivered between continents. In this case, there would be multiple interacting entities of a complete supply chain for end-to-end delivery logistics.
- Mobility: Obviously, problems of mobility are among the most substantial properties of the scenario.
- Evolution of environment: The ability of accommodating change and an evolution of the environment is also one important requirement of the scenario. Beside the essential property of a dynamic extension of the number of participating entities, evolution may include new services, new environments and new technologies.

3.8.5 Feasibility

It can be seen that the basic functions of the scenario can be realized as a proof-of-concept implementation of the CORTEX concepts. The prerequisite for the scenario, autonomous robots will be available and it seems to be feasible to provide functions of a smart environment capable of interacting with the robots. An obvious restriction will be the scale of the scenario. However, it may be possible to demonstrate that there is no system component that really constitutes a bottleneck to extending the scenario.

3.9 Robots in Flare Scenario

Robots in a Flare Scenario may have the following roles:

- providing connectivity of the players in a multi-hop network, thus representing the (mobile) infrastructure
- establishing the global view of the game
- exploring a terrain as a mechanical scout to provide general information to all players

There are also a number of variants that could be introduced depending on the rules of the game chosen:

- having private scouts for special surveillance tasks;
- trying to deactivate adverse scouts.

Providing connectivity of the players in a multi-hop network:

Communication is necessary to disseminate the player's position and provide the link to the virtual part of the game. Because the range of the wireless communication may be restricted, semi-autonomous robots could be used as relay stations to build an ad-hoc multi-hop network connecting all system components. As in the cooperating car scenario, the robots can use position, direction and speed of participants to predict the connectivity and take autonomous actions to maintain it. Actually, the scenario would perhaps be a good model to experimentally explore the characteristics of such a class of applications.

Establishing the global view of the game:

Robots can also be used to provide general information about the position of players and their movement on the virtual game field. The position of players will be provided by the wearable computers of the players that have a location system based on DGPS. The scouts will simply read this position and disseminate or relay it. This task can be done in close relation to the task described above. The other possibility is that players try to hide their position. Then the scouts have to use more sophisticated means of detection. Anyway, the global view of the game changes dynamically and is built up and maintained by a network of scouts. They have to establish a common view that, of course, has to be kept consistent over time. A scout close to a moving participant may have the actual position while the others only have the last updated position and an estimate about the movement derived from the history.

Exploring unknown terrain on behalf of a human player as a mechanical scout:

If a player enters an unknown terrain (one which is not yet modelled as a virtual scenario), robots may explore the area and may jointly create a map. Fixed obstacles and structures have to be distinguished from moving ones. This task is well known from cooperative robotics e.g. RoboCup (the team of the world champion in the middle size league (Freiburg) uses the laser scanners of all robots to create a common view of the field including positions of the other robots and the ball).

The challenge is the temporal correlation of events in the real and the virtual world. If connectivity is lost, and partitions are formed, the respective players have to be alerted.

3.9.1 Autonomy

In general, scout robots may perform tasks in a completely autonomous way. However, there may be the option to use lower levels to get control about fine-grained movements. In this case, the control can be viewed a special case of the remote control scenario.

3.9.2 Consistency

A weak form of consistency may be needed for cooperative actions described above.

3.9.3 Cooperation

Strong cooperation is required for maintaining the network connectivity. Another area of cooperation on the system level is given by the fact that robots may prevent crashing into each other. On the application level, there may be cooperative exploration of a terrain and similar tasks. Because this is not safety or time-critical, soft real-time strategies may be used.

3.9.4 Requirements Analysis

3.9.4.1 Requirements from the environment

- Sentience: Sentience is obviously necessary for the scouts. Because scouts move autonomously in a populated environment, all the properties that are described in section 3.8 also apply here. Additionally, scouts that are responsible for providing

wireless connectivity between the players may have sensors to measure the signal strength as a basis of a location mechanism and a mechanism to prevent partitions.

- Large Scale: The scale of the scenario is closely related to the Flare scenario.
- Safety Critical: As in all scenarios dealing with mobile autonomous vehicles, safety is a major concern. Mechanisms have to be provided to maintain a safe behavior of the scout robots under all circumstances. This includes safety properties of the local system when internal failures occur and safe behavior with respect to the environment. The first property can be derived from the broad body of research in fault-tolerant systems. The second property relates to environment sensing and reliable cooperation protocols.
- Geographical Dispersion: The geographical dispersion of the application is confined by the movements of players and the conditions of the terrain. It is assumed that this will be a locally restricted area.
- Mobility: Obviously, problems of mobility are among the substantial properties of the scenario.
- Evolution of environment: Dynamic extensibility and scalability are important properties of the scenario. Further evolution may be dictated by the rules of the game and, hence, are tightly coupled to Flare.

3.9.5 Feasibility

The feasibility of the scenario is dependent on the terrain in which the robots are to be used. It can be seen that a proof-of-concept application can be realized with a rather restricted number of robots, which are available within the financial frame of CORTEX. These robots may only be partially suited for an outdoor environment

Chapter 4 Summary and Conclusion

This document has presented and analyzed a range of possible applications of CORTEX technology. For the most part these scenarios arise from the scenarios identified in [1] extended to ensure as complete a coverage of the application space as was feasible in the context of this task. In particular, the scenarios are chosen to be representative of applications occupying different points on the spectra of autonomy, consistency and cooperation.

As illustrated in Figure 4.1, the ATTS, teleoperation, and ATC scenarios represent applications with increasing degrees of autonomy of their components.

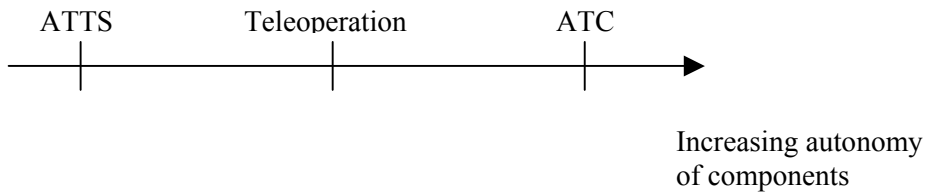


Figure 4.1 Applications requiring different degrees of autonomy

As illustrated in Figure 4.2, the Flare and UIS scenarios represent applications with increasing requirements for consistency.

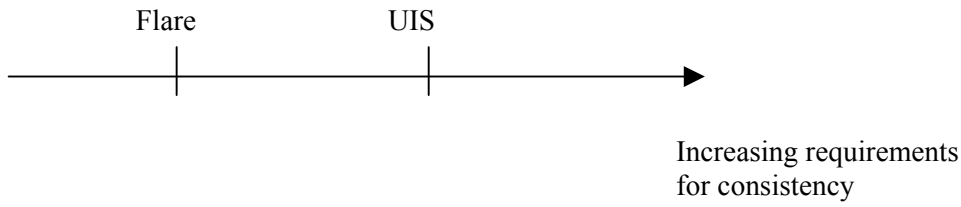


Figure 4.2 Applications requiring different degrees of consistency

Finally, as illustrated in Figure 4.3, the home delivery and cooperating car scenarios represent applications with increasing requirements for cooperation.

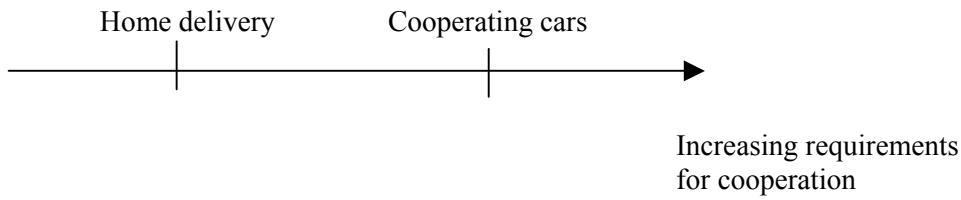


Figure 4.3 Applications requiring different degrees of cooperation

References

[1] CORTEX Annex 1, Description of Work, October 2000.