FAULT TOLERANCE

Pervasive computing envisions an environment that seamlessly integrates digital and physical devices. Users can access digital data and applications from the environment as easily as accessing them through their computers. Since pervasive computing exists in the user's environment, the technology is sustainable if it is invisible to the user and does not intrude the user's consciousness. This requires that functioning of the multitude of devices in the environment be oblivious to the user. Therefore, the system has to be resilient to various kinds of faults and should be able to function despite faults.

Pervasive computing ushers in a new era of computing that integrates digital and physical devices. It envisions a world of computers, handheld devices, sensors and actuators integrated seamlessly with everyday physical devices such as electrical appliances and automobiles. In addition, pervasive computing provides a platform for context-aware computing that enables automatic configuration of a pervasive system based on the environment context. Mark Weiser in his paper “The Computer for the Twenty-First Century” defines pervasive computing as a technology that “weaves itself into the fabric of everyday life until it is indistinguishable from it”.

He mentions that for pervasive computing to be successful, it's functioning should be transparent to the user. Such transparency is achievable if faults in the system are masked and user intervention is sought only when absolutely required. Pervasive computing technology exists in the user’s environment and aids the user in performing various tasks. The sustainability of this technology depends on it being non-intrusive. In order to achieve this goal, faults in a pervasive system should be automatically masked and user notified only when absolutely required.

Fault Tolerance issues have not been well explored so far in pervasive computing research. Since pervasive computing environments operate in the same physical (as well as virtual) space as humans, they can be exasperating (and sometimes hazardous) if they are not resilient to faults. Several researchers have expressed the need for reliable pervasive systems and mention that reliability issues must be readdressed in the realm of pervasive computing. Mentions that one of the paramount concerns of smart home occupants is reliability. Pervasive computing is finding immediate applications in healthcare facilities, aware-homes and assisted-living for the elderly. Sensors are used to monitor conditions of patients in hospitals, onset of age-related disorders in the elderly and status of various electrical appliances in aware-homes. Failures in such scenarios can lead to disasters and so fault tolerance is vital.

The ramifications of faults in a pervasive system can stretch beyond immediate consequences. Faults can lead to incorrect context sensing, security and privacy breaches and misuse of resources. Therefore, fault containment is a very important aspect of deploying a pervasive system into the physical world.

Classification of Failures

A typical pervasive system consists of commercial off-the-shelf (COTS) software and devices whose reliability is not guaranteed. COTS software are sold as “black boxes” and may not be subject to rigid development, verification or testing processes. Interoperability issues further reduce the reliability of a pervasive system. Mobile devices such as handhelds and laptops, with limited battery power, cannot be regarded as totally reliable. Connectivity failures due to devices going out of range or other errors in networks add to faults in a pervasive system. Besides, a pervasive system has a core set of services (like naming, trading, file system, event delivery and discovery and context services) that provide necessary functionality. These services can also fail. Broadly, faults in a pervasive system can be classified into device, application, network and service failures.

1. Device Failures

A pervasive system consists of different kinds of devices such as desktops, laptops, handhelds, sensors, actuators, displays, speakers, scanners, cameras and projectors. Each device has its own set of faults that can potentially contribute to the failure of the pervasive system. Mobile devices, such as laptops and handhelds, have physical constraints such as finite battery power and limited signal strength. So if the battery goes down or if the signal strength is too low they get disconnected from the pervasive system and are regarded as having failed. A more acute problem with devices is when they are alive but operates incorrectly. This is common in faulty sensors and is called a Byzantine failure.

2. Application Failures

Designing reliable software is an expensive process and the cost of debugging, testing and verifying can easily range from 50 to 75 percent of the total development cost. Even in well-tested software systems, bugs of varying severity are found. Pervasive computing includes commercial off-the-shelf applications that may not be well tested. In some situations, applications may work well as stand-alone software but may not inter-operate correctly or reliably with other software. Therefore, pervasive systems should make few reliability assumptions about applications. Application failures include application crashes due to bugs, operating system errors, unhandled exceptions and faulty usage. Pervasive applications are also likely targets for malicious software such as viruses and worms. Viruses and worms cause fail-stop or Byzantine failures.

3. Network Failures

Pervasive systems consist of wired and wireless devices. Therefore, a reliable pervasive system should account for network failures caused by low signal strength, devices going out of range and unavailability of communication channels due to heavy traffic. Network failures lead to unreachable devices that may be wrongly perceived as device failures. Automatic detection of the failure type is an important issue in pervasive computing.

4. Service Failures

As mentioned above, a pervasive system is supported by various services that enable different functionalities. Some of these services are essential while others add features to a pervasive system. Essential services include naming, event and discovery services. Some pervasive systems support other services such as a trading service that enables device discovery, context services that enable context-aware computing and file system services for ubiquitous data access. Examples of service failures include service crashes due to bugs and operating system errors, faulty operation of services like sensing incorrect context, wrong inferring and loss delivery of events. Service failures can potentially lead to failure of the pervasive system.

Implications of Failures

Pervasive computing integrates digital devices seamlessly in our physical environment. Digital devices co-exist with physical devices to aid in accomplishing everyday tasks. Therefore, faults in pervasive systems can be bothersome and result in user annoyance. Consider, for instance, an aware-house that uses radio-frequency badges to identify users. When a user enters the house, the pervasive system identifies him and configures the house to meet his requirements. It adjusts the temperature, turns on his favorite television channel and preferred lights in the house. Failure to correctly identify the person can result in a different configuration and can be a source of annoyance to the user.

Security Vulnerabilities

The ubiquity of deployment of pervasive systems necessitates robust security mechanisms for access control and authentication. Faults can lead to security breaches and consequent compromise of trust. A failed intrusion detection system may not detect an intruder while failure in an authentication system may let users misuse resources and data.

Inferring Incorrect Context

One of the key elements of a pervasive system is context. Context information is used to proactively configure and adapt the environment to meet a user's or group's needs. Various sensors are used to sense context. These include cameras, voice recognition systems, temperature sensors and RF identifiers. Inputs from faulty sensors lead to inaccurate detection of context. A faulty light sensor may erroneously sense the level of natural light in a room to be too low and unnecessarily cause artificial lights in the room to be turned on. In systems that use RF badges for sensing location of people, a faulty RF badge may result in inaccurately determining the location of a person. This could have security implications if location is used to give people certain access control rights.

A bigger challenge in context usage is in inferring context accurately. Various methods are used to fuse data from one or more sensors and infer higher-level contexts. For example, rules could be used to infer the activity in a room. However, these rules could sometimes result in the wrong context being inferred. For instance, a rule such as "If the number of people in a room is more than two and a PowerPoint application is running, then there is a presentation going on in the room" may be correct most of the times. However, there could be situations when it does not hold – for example, if there are two people in the room and one of them is editing a PPT file. Incorrect context can potentially lead to inappropriate resource usage, user annoyance or failure of the pervasive system. In the previous example, incorrectly sensing the editing activity as a presentation, the pervasive system might start a presentation recording application, dim the lights in the room and turn on a microphone to be used by the presenter, which are all inappropriate for the editing activity.

Challenges Facing Fault Tolerance

The area of fault tolerance in computing systems has been enriched through decades of research. Fault tolerance issues have been addressed in various areas of computing systems such as computer architecture, operating systems, distributed systems, mobile6. Prototype Fault-Tolerant Pervasive Computing System We envision a pervasive computing system as a device-rich environment that integrates properties of digital and physical devices seamlessly and refer to it as an Active Space. Our active space is comprised of digital devices such as laptops, plasma displays, handheld devices, finger print scanners, desktop computers, RFID badges and infrared beacons integrated in a physical space. The Gaia meta-operating system provides a set of services to manage the active space. The active space supports context-aware computing through the Gaia Context Infrastructure.

Therefore, a device failure is treated as failure of applications running on that device. This technique tolerates application faults that can be masked by restarting applications. Therefore, the fault model only considers application failures caused by transient errors such as device failures, network faults and failures due to faulty usage. Applications periodically save their states onto a checkpoint storage. The reliability of the storage can be improved by traditional techniques such as RAID and so we do not address its failures.

The fault manager obtains information about the current context from the context infrastructure and it gets device and application properties from the Space Repository. It uses this information to infer a contextually appropriate surrogate device on which the application can be restarted. The failed application is then restarted on the surrogate device using the saved state from the checkpoint storage. This is called rollback recovery and reduces loss of state on failure. The fault management system uses a Prolog based reasoning mechanism to determine the most appropriate surrogate device. It considers parameters such as availability of the device, user preferences and application capability while reasoning. These parameters can be set by the user.