TECHNISCHE UNIVERSITÄT WIEN
VIENNA UNIVERSITY OF TECHNOLOGY

D I P L O M A R B E I T

Secure Internet Interfaces for an Integrated Time-Triggered Architecture

Ausgeführt am Institut für

Technische Informatik – E182
der Technischen Universität Wien

unter der Anleitung von
o. Univ.-Prof. Dr. Hermann Kopetz
und
Dr. Roman Obermaisser
als verantwortlich mitwirkendem Universitätsassistenten

durch

Alexander Schrei
Schopenhauerstraße 72/21, A-1180 Wien
Matr.-Nr.: 9625589

Wien, am 26. August 2007 ______________________

Secure Internet Interfaces for an Integrated Time-Triggered Architecture

During the last years, researchers have been trying to incorporate standard communication protocols, e.g., the widely used family of Internet protocols, into embedded systems. The expected advantages are increased interoperability, ease of training, and the possibility of using commercial off-the-shelf (COTS) components. However, the Internet is designed to be open for every network participant and follows the best-effort principle. This limits its applicability in the area of ultra-dependable embedded systems. On the other hand, recently integrated system architectures combining safety-critical and non safety-critical subsystems into one system for higher dependability at lower hardware effort have been proposed. Both the non safety-critical part and the safety-critical part of such mixed-criticality systems can benefit from standard Internet communication interfaces. Condition-based maintenance, fault diagnosis, and engineering feedback are example applications. While the non safety-critical subsystem can be connected to the Internet via bidirectional interfaces, the safety-critical subsystem can exploit the standard Internet interfaces via unidirectional gateways in order to keep the certification effort as low as possible.

One obstacle for the employment of standard Internet protocols in embedded systems is the fact that protocol security holes and attack strategies have been extensively investigated and the respective information and tools are available to many people. This thesis presents a system model for mixed-criticality systems offering secure Internet communication services. Existing dependability concepts are adopted for security and applied to the system model of the integrated system architecture developed in the course of the European research project DECOS (Dependable Embedded Components and Systems). Vulnerability-scanning software is used and attacks are performed in order to experimentally evaluate the effectiveness of the security services incorporated into a DECOS prototype cluster.

Keywords: dependability, embedded system, security, integrated architecture, DECOS, Time-Triggered Architecture, TTA, Internet, Internet Protocol, IP

Sichere Internet-Schnittstellen für eine integrierte zeitgesteuerte Architektur

Seit einigen Jahren versuchen Wissenschaftler, Standard-Kommunikationsprotokolle, z. B. die weit verbreitete Internet-Protokollfamilie, in eingebettete Systeme einzubinden. Die erwarteten Vorteile sind erhöhte Interoperabilität, vereinfachte Ausbildung und die Möglichkeit, seriengefertigte Komponenten einzusetzen. Allerdings ist das Internet als für jeden Netzwerkteilnehmer offenes System konzipiert, das dem Best-Effort-Prinzip folgt. Dadurch ist seine Anwendbarkeit im Bereich hoch zuverlässiger Systeme eingeschränkt. Andererseits wurden in letzter Zeit integrierte Systemarchitekturen entwickelt, die sicherheitskritische und nicht sicherheitskritische Subsysteme in sich vereinen, um höhere Zuverlässigkeit bei geringerem Hardware-Aufwand zu erreichen. Sowohl der nicht sicherheitskritische als auch der sicherheitskritische Teil solcher „Mixed-Criticality"-Systeme kann von Standard-Internet-Kommunikationsschnittstellen profitieren. Beispiel-Anwendungen sind zustandsabhängige Wartung, Fehlerdiagnose und Engineering-Feedback. Während das nicht sicherheitskritische Subsystem mittels bidirektionaler Schnittstellen an das Internet angebunden werden kann, ist die Nutzung der Standard-Internet-Schnittstellen für das sicherheitskritische Subsystem durch unidirektionale Gateways denkbar, um den Zertifizierungs-Aufwand so gering wie möglich zu halten.

Ein Hindernis für die Verwendung von Standard-Internet-Protokollen in eingebetteten Systemen ist, dass Protokoll-Sicherheitslücken und Angriffsstrategien ausgiebig studiert wurden und die entsprechenden Informationen und Software vielen Menschen zur Verfügung stehen. Diese Arbeit stellt ein Systemmodell für „Mixed-Criticality"-Systeme mit sicheren Internet-Kommunikationsdiensten vor. Vorhandene Zuverlässigkeitskonzepte werden für den Sicherheitsbereich übernommen und auf das Systemmodell der integrierten System-Architektur, die im Rahmen des europäischen Forschungsprojektes DECOS (Dependable Embedded Components and Systems/zuverlässige eingebettete Komponenten und Systeme) entwickelt wird, angewendet. Sicherheitsschwachstellen-Scan-Software wird eingesetzt und Angriffe werden durchgeführt, um die Wirksamkeit der einem DECOS-Prototyp-Cluster hinzugefügten Sicherheitsdienste experimentell zu bewerten.

Schlagwörter: Zuverlässigkeit, eingebettetes System, Sicherheit, integrierte Architektur, DECOS, zeitgesteuerte Architektur, TTA, Internet, Internet Protocol, IP

Acknowledgments

I like to thank Dr. Roman Obermaisser for his very supportive guidance and for sharing his highly valuable scientific experience and knowledge. In addition, I thank Prof. Dr. Hermann Kopetz for his illuminative lectures and scientific works. I also like to thank Roman Benesch, the author of a related thesis dealing with the design and implementation of a TCP software module used in the implementation in the course of this thesis, for the constructive collaboration during the integration of his and my own software. Finally, I thank my parents for supporting me emotionally as well as financially during my studies.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The desire to increase the safety of complex technical systems such as aircrafts, cars, and trains, as well as the wish to enhance plant automation has lead to intense research in the area of hard real-time computer systems. By definition these systems are considered correct only if they meet their temporal specification under all specified load and fault scenarios. The violation of this specification can have catastrophic consequences like injury or even loss of life. A real-time computer system is one part of a larger system—a real-time system. Such a system, e.g., a flying aircraft, consists of human operators, physical components, and a real-time computer system [Kop97]. However, in the following, the term real-time system will be used in short for hard real-time computer system unless otherwise noted.

An example of a temporal specification is a deadline up to which a computation or message transmission has to be completed. The specification of the load to be handled can be something like the maximum number of commands or other external stimuli per unit of time the system must accept without failing. Fault scenarios can be, for example, the disruption of a communication channel by physical conditions or a bug in the software. If such a fault is stated in the system specification to be tolerated, a correct system will by definition not fail even under the occurrence of this fault. Such systems are said to exhibit fault tolerance [ALR01]. The specification of the type and frequency of faults to be handled is called the fault hypothesis [Kop97]. In order to achieve a high probability that the system will provide the specified service for the desired period of time, these assumed faults have to cover the faults occurring in reality with a high probability—the so-called assumption coverage. In addition, the probability that the fault tolerance mechanisms used indeed tolerate the assumed faults must be sufficiently high as well [Pow92].

Furthermore, real-time computer systems are normally embedded systems, i.e., they are part of a larger physical system. For instance, mechanical subsystems of cars can be replaced by computer systems embedded into the car: techniques known as brake-by-wire and steer-by-wire are examples of this approach [DFMP98]. An important characteristic of embedded systems is that they are often mass produced. From this follows that hardware costs are critical and have to be reduced by limitation of performance and memory of the embedded computer systems [Kop97].

In order to reduce hardware resources and to improve coordination of various subsystems of an embedded system, the system can be designed following an integrated approach [OPT07], i.e., various application subsystems, e.g., the braking system and the steering system, are implemented on the same hardware. This is in contrast to the traditional federated approach where application subsystems, are implemented separately on distinct hardware.

At the University of Technology Vienna, a distributed computer system architecture for the construction of hard real-time embedded systems has been developed: the Time-Triggered Architecture (TTA) [KB03]. At the core of this architecture are distributed systems, in which communication takes place by the Time-Triggered Protocol/Class C (TTP/C). These systems provide extensive support for fault-tolerance.

The European research project DECOS (Dependable Embedded Components and Systems) [DEC, OPT07] aims at developing an integrated system architecture which can use the TTA as its base architecture.

1.1 Motivation and Objective

In the last years effort has been spent on the incorporation of wide spread standard technologies into embedded systems [IPR98, KKM+98, CGPS03, Ben02]. Interoperability, ease of training, and the possibility of using commercial off-the-shelf (COTS) components are advantages gained from such an approach [Kuc98, WD03].

This thesis focuses on adding secure Internet communication services to embedded mixed-criticality systems adhering to the integrated DECOS architecture. This is accomplished by supplying the non safety-critical subsystem with Internet protocol stacks [IET89]. These standard Internet interfaces are used only for the implementation of communication services demanding no highly predictable latency and jitter, i.e., they are not used for hard real-time communication. Furthermore, the non safety-critical subsystem can utilize the stacks directly and for bidirectional communication. In contrast, the safety-critical subsystem can exploit the stacks indirectly via unidirectional gateways to the non safety-critical subsystem. Prohibiting information or control flow from the non safety-critical subsystem into the safety-critical subsystem eases the certification of the safety-critical subsystem [KOPS04]. The standard Internet communication services are established on top of the core time-triggered communication service (for system internal communication) and the Ethernet protocol (for external communication). The use of standard Internet protocols introduces additional security risks because these protocols are well known and methods of misusing these protocols have been studied extensively [Gar00, HM96, Mar00, KaC00]. In addition, the possible connection of an Internet enabled embedded system to the Internet poses a security risk. As a consequence, security mechanisms complementing the TCP/IP stacks are presented.

Although many important communication tasks in the context of real-time systems demand for a hard real-time communication interface, i.e., an interface having a precise temporal specification and meeting this specification under all specified load and fault scenarios, several exceptions can be identified [Kop00]. [Kop00] distinguishes between the real-time service (RS) interface, the diagnostic and management (DM) interface, and the configuration planning (CP) interface of a component being part of a real-time system. Normally, only the first of these three interfaces is required to have a precise temporal specification and to meet this specification under all specified load and fault conditions. The second and the third normally can be implemented according to the best-effort paradigm [Kop97].

In the following, detailed motivation for complementing the hard real-time communication services of a real-time system with non hard real-time communication services, i.e., soft real-time or non real-time communication services, is given. Furthermore, arguments are presented why the standard Internet protocols in many cases are a good choice for the provision of non hard real-time communication services.

Complementing Hard Real-Time Communication Services with Non Hard Real-Time Communication Services

Example cases in which an additional non hard real-time communication interface can be beneficial are presented below:

- Development:
  - Testing: Testing is the process of verifying experimentally that a system adheres to given properties [ALR01]. Normally, these properties

    are derived from the specification. In an approach similar to the one presented in [Hex03], tests can be conducted automatically using a non hard real-time interface: a series of tests of the real-time system is started by an external workstation, which in turn retrieves the results for evaluation.

  - Fault Diagnosis: Fault diagnosis is the process of identifying and recording the cause(s) of error(s), in terms of both location and type [ALR01]. [SW01] elaborates on an approach of debugging CORBA objects remotely. In order to apply such an approach to real-time systems, a non hard real-time communication interface can be more efficient.

- Maintenance:

  - Fault Diagnosis: See above.
  - Support of Condition-Based Maintenance by Gathering of Run-Time Information (Just-in-Time Maintenance): For example, the discrete Fourier analysis of the vibrations of a train engine can help to approximate the point in time of a wearout failure. This technique is superior to periodic preventive maintenance or repair after failure [IPR98, SLG$^+$01].
  - Software Updates [Pfe01]: Communication interfaces for the update of software are normally not hard real-time and have to be provided in many real-time systems.

- Operation:

  - Configuration [HS99]: System parameters, e.g., product quality ranges in plant automation, can be set before the actual real-time service is started.
  - Mixed-Criticality Systems: For economic and dependability reasons, safety-critical and non safety-critical parts of a system can be implemented to share the same resources, e.g., the same hardware node. In such mixed criticality systems [HL00, OPT07], at least for the non safety-critical subsystem a non hard real-time interface might be more efficient since one communication channel can be shared by several applications in a non-deterministic way according to current demands.

Complementing Hard Real-Time Communication Services with Remote Non Hard Real-Time Communication Services

Again focusing on non hard real-time communication, in the following, advantages to be expected from remote communication capabilities are pointed out [IPR98]:

- Fewer or no operators have to be on site because one operator can communicate with several real-time systems at different locations.
- Interruptions of service due to maintenance activities are shortened: In case remote maintenance is possible, fewer or shorter maintenance stops, e.g., of trains, are necessary. Fault diagnosis for cars can be conducted remotely.
- Expert knowledge at remote sites can be utilized.
- The tools and replacement parts necessary for repair or maintenance can be determined before arrival at the site.
- Large field tests can be conducted by collecting information from systems currently in use.

Arguments for the Use of Standard Internet Protocols

Since the Internet protocols enable both local and remote communication, advantages mentioned in both of the two sections above can be gained simultaneously by employing these protocols. In addition, the following benefits can be expected:

- Interoperability:
  - Due to standards like HTML and Java, platform independent client applications can be employed [JCH00].
  - [HS99] mentions the problem of system solutions being too dependent on low-level communication protocols. Internet protocols can be used on top of many low-level communication technologies.
  - Links to information on the Internet, e.g., system documentation, can be provided and combined with the Web interfaces to the real-time system [NFW01].
- Using of COTS Components:
  - The client application does not have to be installed on the external workstation because one part of the client application is mostly installed

already (a Web browser) and the other part (the HTML code and/or Java applet) is downloaded from the server being a part of the real-time system [IPR98].

- No additional private network has to be installed because the Internet can be used [NFW01].
- Wide spread and sophisticated Internet technologies such as HTTP, HTML, SSH, and Java can be used more easily.

• Ease of Maintenance:

  - In many cases one point upgrade suffices: For example, if a simple HTML server is used, software versions on the device do not have to match precisely the versions at the client, as long as an HTML version compatible with the browser is used [JCH00].
  - Basic Internet protocols such as IP (Internet Protocol) seem to adhere to a low pace of change: The official standard for IP version 4 [USC81a] has been published in 1981 and the protocol is still in use. The first standard for its successor, IP version 6, from the year 1995 [DH95] has only reached the highest but one of three standard maturity levels, yet [DH98]. These levels are from lowest to highest "Proposed Standard", "Draft Standard", and "Standard" [Bra96].

1.2 Structure of this Thesis

Chapter 2 presents the concept of dependability, the threats to dependability—faults, errors, and failures—and techniques for attaining dependability—fault prevention, fault tolerance, fault removal, and fault forecasting. After a comparison of event-triggered and time-triggered systems, the concepts of federated and integrated architectures are presented. Next, the integrated DECOS architecture and the TTA are introduced. Then, the Internet protocol family, the domain of "Embedded Internet", and the concept of gateways is presented. Finally, a security fault model, prominent techniques for attaining security, and the topic of "Embedded Security" are illustrated.

Chapter 3 discusses related research considering the aspects of varying interconnection architectures, and approaches to embedded security.

Chapter 4 presents the system model from low-level to high-level communication services. Next, the dependability model with respect to non-malicious faults and malicious faults is outlined. Important cases of system external and system internal security faults are discussed. Finally, possible applications of the system model classified into development, maintenance, and operational applications are illustrated.

Chapter 5 describes the DECOS prototype cluster, the implementation of the virtual TCP/IP networks, and the implementation of the security gateway.

Chapter 6 presents a theoretical evaluation of the system model and a theoretical and experimental evaluation of the implementation.

Chapter 7 concludes the thesis and provides an outlook on possible future work.

Chapter 2

Fundamentals

2.1 Dependable Real-Time Systems

Computer systems and components vary in the degree they can be depended on. In order to be considered dependable a system has to fulfill a number of attributes. One important attribute is availability, i.e., the system provides correct service for at least an a priori specified percentage of an observed time interval. For instance, a car's navigation system should be operational and provide correct service at such a high percentage of the time it is used that it is economical feasible. Another attribute of dependability is safety. It can be measured by the mean time to a catastrophic failure, i.e., a failure having catastrophic consequences for humans or the environment. Computer systems controlling safety-critical functions of aircrafts, cars, trains, etc. must be safe to a very high degree.

2.1.1 Dependability

[ALR01] defines dependability of a computing system as the ability to deliver service that can justifiably be trusted. A system failure is the occurrence of a deviation of the delivered service from correct service, i.e., the intended service. The period of incorrect service delivery is the service outage [ALRL04]. Service restoration is the transition from incorrect service to correct service. A system error is the part of the system state that may cause a subsequent failure. The adjudged or hypothesized cause of an error is a fault. The definition as adjudged or hypothesized intends to stop recursion in the determination of a cause. For instance, following this definition, a programmer's mistake can be defined as a fault without necessarily searching for the causes for the programmer's mistake. Faults and errors are states (the state of an object is the collection of the values of all properties of an object at a point in time), while failures are events (an event is an occurrence at a point in time) [Kop97].

Errors can propagate from system to system (e.g., by erroneous messages) as outlined in Figure 2.1. When the error becomes visible (half square in Figure 2.1) at the service provider's service interface the service provider fails. From the service consumer's point of view, this error can be interpreted as an external fault (half circle in Figure 2.1). The causal chain of fault, error, and failure is illustrated in Figure 2.2. A fault may be located inside the system itself (internal fault, see Figure 2.3 (a)) as in the case of a software bug or external to the system (external fault, see Figure 2.3 (b)) as in the case of a wrong input by an operator.

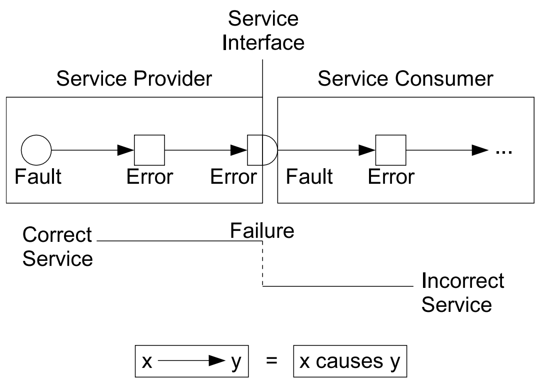


Figure 2.1: Error Propagation Across Subsystem Boundaries (following [ALRL04])

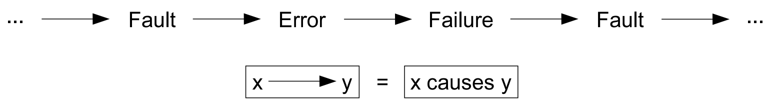


Figure 2.2: The Causal Chain of Fault, Error, and Failure (following [ALRL04])

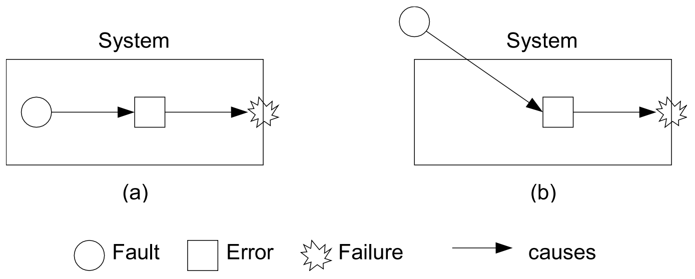


Figure 2.3: Internal versus External Faults (following [ALRL04])

Dependability is the superordinate concept of the following attributes [ALR01]:

- Availability: “readiness for correct service” [ALR01]

  Availability can be measured by the percentage of correct service delivery with respect to the overall time of correct and incorrect service delivery.

- Reliability: “continuity of correct service” [ALR01]

  The longer the system provides correct service continuously after a point of time of correct service delivery, the higher is its reliability.

  In the context of hard real-time systems, reliability is an important attribute because even short service outages might have catastrophic consequences. For example, an unexpected delay in the braking system of a car might lead to an accident.

- Safety: “absence of catastrophic consequences on the user(s) and the environment” [ALR01]

  In the case of transportation systems, the safety of humans transported and the environment has to be guaranteed with a high probability.

- Confidentiality: “absence of unauthorized disclosure of information” [ALR01]

  In recent years, the security risks of the Internet have lead to increased awareness of confidentiality issues.

- Integrity: “absence of improper system state alterations” [ALR01]

  Integrity is an important attribute of dependability both in the security context and in the context of non-malicious faults. Due to decreasing sizes of semiconductor devices the disturbing impact of manufacturing residuals, alpha particles originating from radioactive impurities in device materials, and high-energy neutrons from cosmic radiation is increasing. Soft permanent faults [Kop04], i.e., the corruption of the component’s state without causing permanent damage to the component, might be the result [Bau01, Con02].

- Maintainability: “ability to undergo repairs and modifications” [ALR01]

  Since system errors and system evolution cannot be avoided completely, repair and modification procedures must be short and cheap.

Four techniques for attaining dependability can be identified [ALR01]:

- Fault prevention: “how to prevent the occurrence or introduction of faults” [ALR01]

Fault prevention includes techniques like rigorous design, physical shielding, and user training.

- Fault tolerance: "how to deliver correct service in the presence of faults" [ALR01]

  Fault tolerance is generally implemented by error detection and subsequent recovery. Recovery is done by error handling and fault handling. Three types of error handling exist:

  - Rollback: A previous correct state is used as a point of continuation.
  - Compensation: The error is masked by exploiting redundancy contained in the erroneous state.
  - Rollforward: A new correct state is established and used as a point of continuation.

  Fault handling is a procedure involving four steps:

  - Fault diagnosis: identification and recording of the cause(s) of error(s), in terms of both location and type
  - Fault isolation: prevents influence of the fault on the system service (e.g., deactivation of the faulty component)
  - System reconfiguration: reassignment of tasks to non-failed components
  - System reinitialization: checking, updating, and recording the new configuration, updating system tables and records

  Fault masking is systematic use of compensation without explicit error detection, e.g., replication of physical connections without error detection mechanisms. However, in practice additional error detection and possibly fault handling is advisable in order to avoid progressive loss of protective redundancy.

- Fault removal: "how to reduce the number or severity of faults" [ALR01]

  During the development phase, the system specification and the system itself can be checked in order to eliminate faults. Checking the specification is termed validation while checking the system is referred to as verification. During the operational phase, fault removal is preventive or corrective maintenance. Preventive maintenance removes faults before they cause errors whereas corrective maintenance removes faults after they have caused errors.

- Fault forecasting: “how to estimate the present number, the future incidence, and the likely consequences of faults” [ALR01]

  Fault forecasting is the evaluation of the system with respect to faults and their consequences. Qualitative evaluation classifies and ranks event combinations (considering the system and the environment) that might lead to system failure. Quantitative evaluation aims to determine the probabilities at which some of the attributes of dependability are satisfied.

2.1.2 Event-Triggered versus Time-Triggered Systems

Real-time systems can be designed to be event-triggered (ET) or time-triggered (TT) [Kop97]. In ET systems, computations or message transmissions are initiated by the occurrence of events, e.g., the reception of a message or a change of the controlled object’s state. In TT systems, actions are initiated at predetermined points in time. TT schedules are mostly periodic, i.e., a sequence of actions is executed over and over again. Many control applications can be implemented in a TT fashion: at first, sensors gather information about the controlled object’s state, then the actuators are adjusted to appropriate values in order to bring the controlled object closer to the desired state, then the sensor values are read again, and so on.

One of the main advantages of TT systems is predictability: in ET systems, actions can be initiated at any time and thus a vast number of sequences of actions is possible; in TT systems only the predetermined sequence of actions can occur. This eases the validation of the system’s temporal behavior and allows for the implementation of error detection mechanisms based on a priori knowledge (e.g., message arrival times). The main disadvantage of the TT approach is the inability to adapt to varying loads: due to the use of static schedules in strictly periodic systems, the computational load cannot be balanced on demand. [Kop97] points out the need for such kind of flexibility in real-time systems: for instance, an emergency shutdown message has to be serviced within a very short latency. A possible solution for TT systems is the reservation of communication and processor resources for this service in every control cycle. However, such a message might be received only very seldom making such a solution inefficient in resource utilization. Another approach providing better resource utilization is the implementation of mode changes: the schedule is exchanged at run-time in order to adapt to varying scenarios.

2.1.3 Event Messages versus State Messages

In distributed real-time systems, information has to be exchanged between the system components by the transmission of messages. Two different basic types of messages can be identified: event messages and state messages. For example, in order to communicate the current desired velocity of a car from the accelerator pedal to the engine at least two approaches are possible. An event message can be sent every time the position of the pedal changes containing the amount of change or on the other hand, a state message can be sent at regular intervals (e.g., once a millisecond) containing the current position of the pedal.

In [Kop97, KS03] the terms event information, state information, event message, and state message are introduced (Tables 2.1 and 2.2). Event information contains information about an occurrence at a point in time while state information contains information about the state of an object at a point in time. Event information must not be lost or duplicated in order to allow the calculation of the current state of the object (exactly-once semantics). State information must not be lost but it can be duplicated, i.e., it is idempotent (at-least-once semantics). Event information has to be queued on sending and receiving in order to compensate for the varying frequency of event occurrences while state information can be updated (at the sender and the receiver) by overwriting.

	Event Information	State Information
Content	information about an event, i.e., an occurrence at a point in time	information about the state of an object at a point in time
Semantics when processed at the receiver	exactly-once semantics	at-least-once semantics (idempotent)
Queuing at sender	queued, consumed on sending	update-in-place (overwrite), not consumed on sending
Queuing at receiver	queued, consumed on reading	update-in-place (overwrite), not consumed on reading

Table 2.1: Event Information versus State Information

Event messages carry event information and state messages carry state information. The sending of event messages is controlled by the sending component (external control) while the sending of state messages is controlled autonomously by the communication system (autonomous control). Normally, state messages are sent in a time-triggered fashion, e.g., periodically. In the case of event messages, the receiver has to signal overloading to the sender (back-pressure flow control). In the case of state messages, the receiver and the sender agree a priori (e.g., by the definition of the message schedule) how many messages will be communicated within a specified time interval. Due to the use of back-pressure flow control, event message flow control is bidirectional while state message flow control is unidirectional. The detection of event message transmission errors is accomplished at the sender by the use of a time-out for the acknowledgment message. At the receiver, transmission errors cannot be detected if messages are sent only whenever an event occurs, because the receiver cannot distinguish between an interval without events and an interval of disruption of the communication channel. State message transmission errors can be detected at the receiver by the use of a priori knowledge, e.g., the transmission schedule. However, in addition acknowledgments can be used for error detection at the sender.

	Event Messages	State Messages
Content	event information	state information
Control strategy	external control (event-triggered, explicit flow control (back-pressure flow control))	autonomous control (normally time-triggered, implicit flow control (unidirectional))
Error Detection	at sender based on time-out for acknowledgment	at receiver based on a priori knowledge

Table 2.2: Event Messages versus State Messages

2.1.4 Federated versus Integrated Architectures

[OPT07] differentiates between federated and integrated architectures for dependable embedded systems. A federated architecture employs a dedicated computer system for each application subsystem. For example, the braking system, the comfort system (heating, seat adjustment, etc.), and the multimedia system of a car might be assigned to three distinct distributed computer systems. In contrast, in an integrated architecture, all application subsystems are assigned to various parts of one distributed computer system.

The advantages of federated systems are natural fault containment and natural complexity management due to the physical separation of application subsystems. However, integrated systems demand for fewer physical resources and ease tactic coordination of application subsystems. As a consequence, costs are reduced (due to fewer physical resources) while dependability can be increased, e.g., by reduction of wiring and connectors. An integrated architecture trying to combine the advantages of both approaches is the DECOS architecture described in the following section.

2.2 The DECOS Architecture

In the course of the European research project DECOS (Dependable Embedded Components and Systems) [DEC, OPT07], an integrated architecture for dependable embedded systems is developed. In this architecture, the application subsystems such as the steer-by-wire, brake-by-wire [DFMP98], or the multimedia subsystem of a car are denoted as Distributed Application Subsystems (DASs). During the requirement analysis of the application, the system is decomposed into nearly independent DASs. In turn, every DAS is decomposed into several jobs. Later, these jobs are assigned to various partitions of the hardware node components of the distributed system, in which they are executed in isolation from the other jobs of the same node. [OPK05] distinguishes between the functional and the physical system structure as illustrated below.

2.2.1 System Structure

As the name implies, normally a DAS is implemented in a distributed fashion on several nodes and one node contains several jobs (software subsystems) of different DASs.

The physical system structure (Figure 2.4) provides partitions for the jobs. These partitions are strongly isolated from each other by architectural mechanisms. As a consequence, advantageous properties of the federated approach such as fault containment and intellectual property protection are not lost in spite of the integration. The hardware node components communicate with each other by a physical network adhering to the time-triggered paradigm.

The functional system structure (Figure 2.5) consists of DASs, virtual networks built on top of the physical network, and jobs. The virtual networks are established by statically assigning sending slots of the physical network to virtual networks. As a consequence, DASs remain independent although they are sharing the same physical resources. While jobs are assigned to partitions, virtual networks are assigned to sending slots of the physical network.

The DECOS architecture provides a set of core services and an extensible set of high-level services built on top of the core services. The architectural services

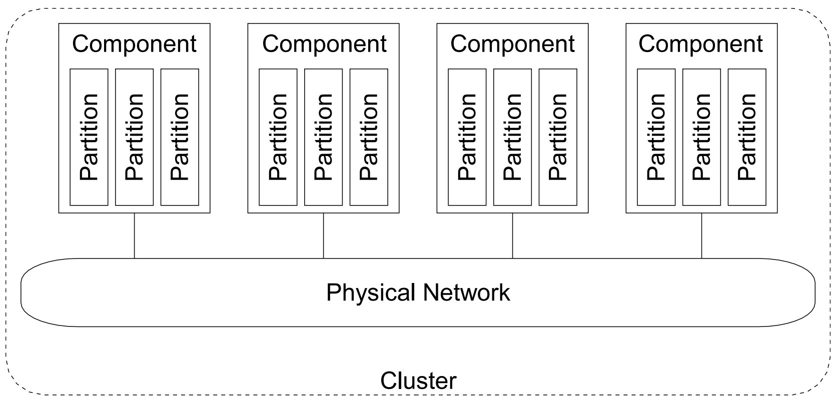


Figure 2.4: DECOS Physical System Structure (following [OPK05])

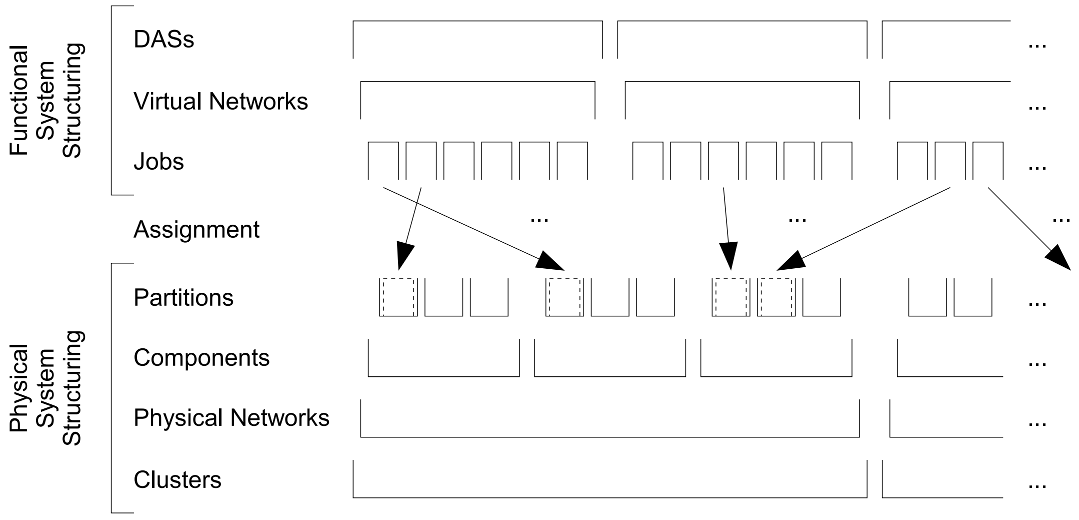


Figure 2.5: DECOS Physical vs. Functional System Structure (following [OPK05])

separate the application functionality from the platform technology. These services hide implementation details from the application while providing to it enough information allowing the satisfaction of application requirements such as timeliness and dependability.

2.2.2 Core Services

In the following, the DECOS core services are presented. An example architecture providing these core services is the Time-Triggered Architecture (TTA) described further below. [OPT07] defines three core services: deterministic and timely transport of messages, fault-tolerant clock synchronization, and strong fault isolation.

Deterministic and Timely Transport of Messages

This service is provided by employing a physical replicated time-triggered network. A Time Division Multiple Access (TDMA) [Kop97] scheme controls the periodic time-triggered state message transmissions, i.e., nodes send only at reserved time-slots defined before run-time in order to preclude collisions.

Fault-Tolerant Clock Synchronization

In order to compensate for the diversity of the physical clocks of the distributed components, clock synchronization has to be employed.

Strong Fault Isolation

[Kop03] defines a fault-containment region (FCR) as a set of subsystems that share one or more common resources and may be affected by a single fault. Common resources can be computing hardware, power supply, software modules, etc. [Kop04, OP06]. In order to prevent errors occurring in an FCR from propagating to other components of the system (e.g., by erroneous messages), error-containment regions (ECRs) can be established. An ECR contains at least two independent FCRs, i.e., it contains enough redundancy (one or more components not directly affected by the fault) in order to compensate for the fault and isolate the errors. The errors cannot propagate outside the boundaries of the ECR [KOPS04].

In addition, the DECOS architecture ensures that a message is either consistently received by all correct components or detected as erroneous by all correct components.

2.2.3 High-Level Services

[KOPS04] presents five generic high-level services: the encapsulation, the virtual network, the hidden gateway, the fault tolerance, and the diagnostic service.

Encapsulation Service

The encapsulation service provides temporal and spatial error containment. In order to inhibit unintentional interactions between jobs contained in one node, each job is constrained to a protected partition and may access component resources (e.g., memory and processor) only to the extent allowed by the encapsulation service.

Furthermore, the encapsulation service statically allocates component resources to the safety-critical and the non safety-critical part of the system in order to ease certification of the safety-critical part independent from the other parts of the system. Moreover, information or control flow is allowed to occur only from the safety-critical subsystem to the non safety-critical subsystem but not in the other direction.

Virtual Network Service

On top of the time-triggered communication service, virtual network services are built: certain data bytes of the time-triggered messages are statically allocated to a virtual network service. In this way, several virtual networks can be implemented using only one physical network while ensuring the temporal separation of these networks. Each DAS is equipped with a dedicated virtual network. Due to the advantages of time-triggered communication (see sections 2.1 and 2.3.1) only time-triggered virtual networks are used for safety-critical DASs.

For non safety-critical DASs event-triggered virtual networks can be employed. In order to enable ET communication on top of the TT communication, Event-Triggered Communication Channels (ETCCs) on top of the Time-Triggered Protocol are introduced: event messages are split into a sequence of small packets, which are transmitted in parts of TT sending slots reserved for ET communication. In order to avoid error propagation from the receiver to the sender, ETCCs employ only unidirectional control flow, i.e., a buffer overflow at the receiver leads to a message omission failure. These failures can be avoided by agreeing a priori on message sizes and message transmission rates, i.e., employing an implicit flow control scheme, or using explicit flow control (e.g., back-pressure flow control) at a higher protocol layer. ETCCs can be used not only for CAN communication (see the paragraph below) but also for many other ET protocols, e.g., TCP/IP as done in the context of this thesis. ET virtual networks are composed of several ETCCs, i.e., unidirectional multicast (one-to-many) communication channels.

[Obe02] presents an approach of emulating a Controller Area Network (CAN) communication controller in a TT system such as a TTP/C system. The CAN protocol is used for control applications in the automotive industry, as well as in other fields. In contrast to TTP/C, CAN adheres to the ET paradigm, i.e., messages are sent on demand at arbitrary points in time. Collisions on the bus are avoided by message priorities: if two or more nodes start sending at the same point in time, only the node sending the message with the highest priority is allowed to continue sending while the others have to abort transmitting.

In analogy to ET virtual networks, a TT virtual network can be interpreted as the composition of several unidirectional multicast (one-to-many) communication channels denoted as Time-Triggered Communication Channels (TTCCs). Table 2.3 outlines the common and differing properties of TTCCs and ETCCs. Both offer multicast communication and the dynamic specification of message content. While in the case of TTCCs, the message lengths and send instants are defined at design time, in the case of ETCCs, these properties can be set dynamically for every message at run-time. Messages transmitted via TTCC are not queued while those transmitted via ETCCs are queued. Flow control can be implicit (in both cases) or (in the case of ETCCs) implemented at a higher layer.

Property	TTCCs	ETCCs
Multicast	yes	yes
Content	dynamic	dynamic
Message Length	static	dynamic
Send Instant	static	dynamic
Queuing	no (overwrite)	yes
Flow Control	implicit (unidirectional)	implemented at higher layer or implicit (unidirectional)

Table 2.3: TTCCs versus ETCCs

Hidden Gateway Service

Since every virtual network is assigned to only one DAS, for communication between different DASs gateways need to be employed. Gateways implemented at the architectural level (i.e., transparent to the application) and connecting virtual networks are called hidden virtual gateways. Gateways allow controlled sharing of resources and coordination between DASs. By selectively redirecting only a subset of messages from one virtual network to another, the complexity of the system is reduced.

In contrast to hidden virtual gateways, hidden physical gateways are employed for the connection of different physical networks, e.g., the connection of a field bus network to the core real-time network. A second application of hidden physical gateways is the reuse of legacy systems: for example, a CAN system can be integrated into a DECOS system.

Fault Tolerance Service

The fault tolerance service provides error compensation (see section 2.1.1) by the cooperation of replicated jobs physically distributed across the system. For example, the output of three jobs providing the same service (triple modular redundancy (TMR) [Kop97]) can be used as input to a voting mechanism choosing the output produced by the majority of the jobs.

Diagnostic Service

The diagnostic service consists of a diagnostic acquisition service and a diagnostic dissemination service. The acquisition service collects data such as the identity of disagreeing jobs in a TMR configuration. The dissemination service enables producers of diagnostic information to send this information without the need for back-pressure flow control by using the ETCCs introduced in section *Virtual Network Service* above. In this way, error propagation from the diagnosis subsystem to the safety-critical subsystem is precluded in the case of diagnostic information being sent from a safety-critical DAS to the diagnosis subsystem. This eliminates the need for certification of the diagnosis subsystem to the highest level of criticality.

2.2.4 Component Structure

Figure 2.6 depicts the basic internal structure of a DECOS node. The communication network is connected to the jobs by connector units. These are software and/or hardware modules forwarding messages between modules, e.g., the communication controller and higher layer connector units. The connector units provide network resource allocation and fault-tolerance mechanisms such as voting on replicated messages. The interface between the basic connector unit and the communication controller is called the communication network interface (CNI). The basic connector unit transfers data to be sent to the CNI. Then, the communication controller retrieves the data from the CNI and sends it to the network at a predetermined point in time. The points in time of sending are chosen at design time adhering to the TDMA scheme. At the receiving node, the communication controller transfers the received data to the CNI from which it is retrieved in turn by the receiving basic connector unit at some specified time later. In this way, the communication controller and the CNI act as a temporal firewall [KS03] between the communication network and the basic connector unit: since also the *amount* of data sent each time (the message length) is specified at compile time, the basic connector unit cannot consume more bandwidth than dictated by the schedule and on the other hand the communication network cannot overload the basic connector units with received messages.

Data transfer between the complex connector unit and jobs of the non safety-critical subsystem might be time-triggered or event-triggered. In the safety-critical subsystem data transfers are time-triggered in order to take advantage of increased predictability.

The basic connector unit statically allocates network resources to the safety-critical and the non safety-critical subsystem. The safety-critical connector unit statically allocates network resources to the various jobs of the safety-critical subsystem while the complex connector unit statically allocates network resources to the jobs of the non safety-critical subsystem.

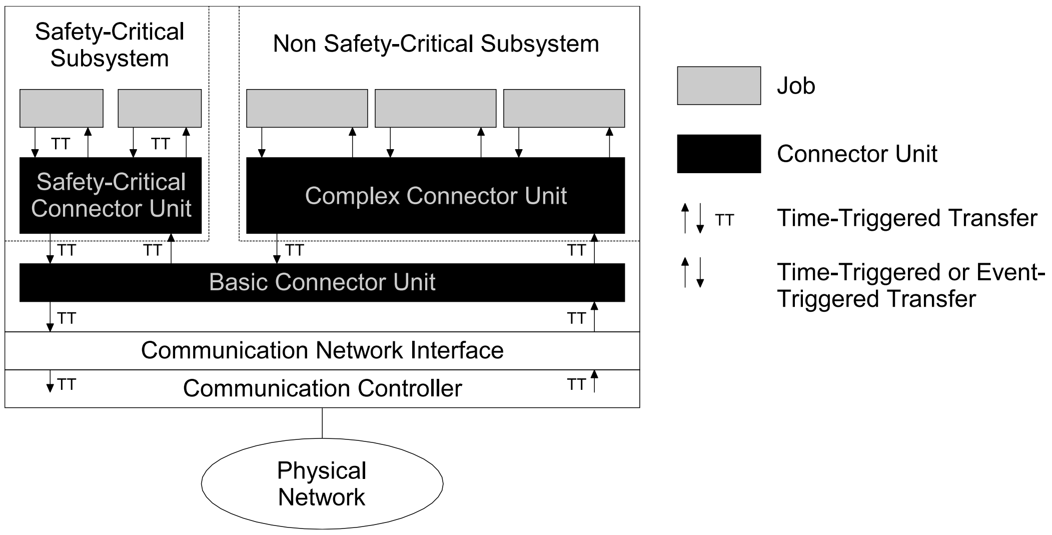

Safety-Critical Subsystem
Non Safety-Critical Subsystem
TT
Safety-Critical Connector Unit
Complex Connector Unit
Basic Connector Unit
Communication Network Interface
Communication Controller
Physical Network
Job
Connector Unit
Time-Triggered Transfer
Time-Triggered or Event-Triggered Transfer

Figure 2.6: Internal Structure of a DECOS Node (following [KOPS04])

2.3 The Time-Triggered Architecture

The Time-Triggered Architecture (TTA) [KB03]—developed at the University of Technology Vienna—is a system architecture for the construction of TT hard real-time systems. It can be used as a base architecture for a DECOS system. A TTA system might consist of two kinds of subsystems (Figure 2.7): real-time clusters, in which the computational nodes communicate by the Time-Triggered Protocol/Class C (TTP/C) [KG94], and field bus clusters, in which communication takes place by the Time-Triggered Protocol/Class A (TTP/A) [KHE00]. While TTP/C provides extensive support for fault tolerance, TTP/A is a scaled down version in order to reduce costs at the field bus level. Normally, the TTP/A nodes are sensors or actuators containing a microcontroller. The two independent TTP/C networks for the real-time clusters offer extended protection against communication failures.

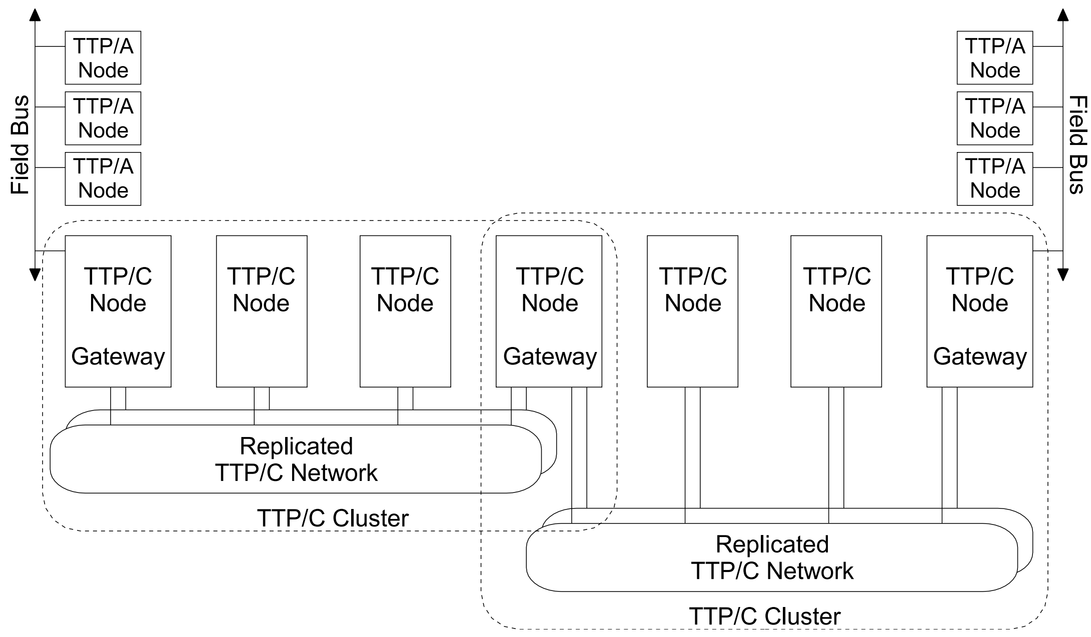


Figure 2.7: Example TTA System Structure

2.3.1 DECOS Core Services

The TTA provides the core services demanded for the integrated DECOS architecture by employing its communication protocol TTP/C. The TTP/C protocol exploits TT communication and the a priori knowledge due to the static schedules in several ways:

Deterministic and Timely Transport of Messages

Unlike many event-triggered protocols, TTP/C does not include a retry mechanism (message retransmission in case of failure). This leads to small communication jitter (i.e., the difference between the maximum duration of a communication action and the minimum duration of a specified set of possible durations [Kop97]) and high determinism.

Fault-Tolerant Clock Synchronization

Due to the a priori known instants of message transmissions, the actual points in time of the transmissions can be used as input to a fault-tolerant clock synchronization algorithm. In this way the clocks of the individual nodes are kept in close agreement with a known precision and a globally synchronized notion of time, the so-called global time, can be implemented [Kop97].

Sparse Time Base: A sparse time base [Kop97, KB03] is established by agreeing a priori that a set of events (e.g., message transmissions) is only allowed to occur during periodic intervals of "activity" separated by periodic intervals of "silence". The duration of these intervals is chosen in order to enable the conclusion about the sequence of events that occurred during distinct intervals of activity. Events that occur during the same interval of activity are said to be concurrent. As a consequence, no agreement protocol establishing a temporal order of these events needs to be executed. Only events not conforming to the sparse time base, e.g., events that are not in the computer system's sphere of control, have to be ordered using an agreement protocol.

Replica Determinism: Due to the implementation of a sparse time base, two or more nodes can more easily come to a consistent view of the current state of the system. An algorithm using the current view of the system's state as input can be designed to produce the same results on all nodes at approximately the same points in time. If two or more nodes visit the same externally visible state and produce the same output messages at points in time at most d time units apart, the nodes are said to be replica determinate [KB03]. This property is important for the provision of fault tolerance by replication of nodes.

Strong Fault Isolation

A TTP/C physical network provides fault isolation by use of star couplers also acting as guardians. As depicted in Figure 2.8, they connect the TTP/C communication controller to the actual communication network and protect the network from messages sent outside the specified time slots.

The TTP/C protocol provides a membership service, i.e., every node keeps a consistent vector of node identifiers of nodes sending correctly.

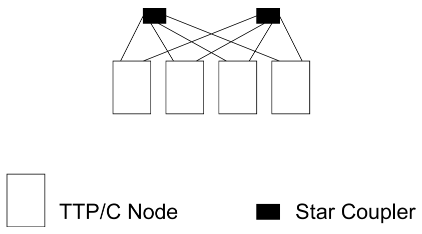


Figure 2.8: TTP/C Star Couplers (following [KB03])

2.4 The Internet Protocols

In recent years, the Internet has grown significantly [Int] and has become one of the world's major communication networks. Despite its size, its underlying principles are simple: According to [IET89], the Internet is a network of interconnected computer networks as depicted in Figure 2.9. A local network is a network of a number of hosts or routers often located closely to each other. A host is a communication end point enabling users to communicate with other hosts on the Internet. A router is a device connected to two or more local networks forwarding data packets between these networks. In this way, data packets can traverse the Internet from one host to another—possibly remote—host by passing through zero, one, or multiple routers.

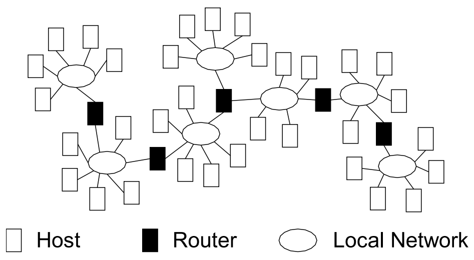


Figure 2.9: The Basic Structure of the Internet

The Internet communication protocols are organized in four layers as illustrated in Figure 2.10. Each layer is meant to perform a part of the communication task:

- The link layer is responsible for communication in the local network only. A very common example protocol is the Ethernet protocol. A supporting protocol for other link level protocols is the Address Resolution Protocol (ARP) [Plu82]: it resolves Internet layer addresses (e.g., IP addresses (see below)) to link layer addresses (e.g., Ethernet addresses) by a simple query/reply scheme.

- The Internet layer's main protocol is the so-called Internet Protocol (IP)

[USC81a], which is responsible for delivering IP datagrams—the data packets mentioned above—from the source to the destination. This is accomplished by assigning at least one globally unique numeric IP address to every host. The routers maintain information about how to find a short path to a specified IP destination address. The IP protocol offers no guarantee for the delivery of a datagram or the order of delivery of a sequence of datagrams. Even the corruption of the datagram's data or the duplication of a datagram sent only once are no violations of the IP specification.

The Internet Control Message Protocol (ICMP) [Pos81] is used for supporting tasks like error reporting, e.g., in case an IP address is currently not reachable.

- The transport layer makes use of IP's datagram service for the provision of host-to-host communication. Both, the Transmission Control Protocol (TCP) [USC81b] and the User Datagram Protocol (UDP) [Pos80] employ checksums for the detection of data corruption. Moreover, they provide a refinement of the Internet address space by the introduction of ports: these are two-byte numbers used as a sub-address within the host in order to demultiplex received data to the various applications concurrently communicating by TCP or UDP.

  In addition to these services, TCP provides sequence numbering and acknowledgment techniques in order to provide a—to some extent—reliable, bi-directional communication path for ordered byte streams. In this way, arbitrary long data streams can be sent across the Internet by letting TCP split them up into small segments sent as IP data automatically.

- The application layer provides more specialized services than the transport layer. For example, the Hypertext Transfer Protocol (HTTP) [FGM+99] is the underlying communication protocol for downloading Web pages from servers on the Internet. The Domain Name System (DNS) protocol [Moc87a, Moc87b] helps to resolve character string names of hosts (domain names) as, e.g., `www.isc.org`, to their numeric IP addresses. While HTTP communicates on top of TCP, the DNS protocol uses UDP as its transport protocol.

The packet format of most protocols consists of a header containing protocol-specific data like the destination address, a checksum, sequence numbers, etc., and a data portion containing upper-layer packets or user data. For instance, IP (version 4) datagrams consist of a 20 to 60 byte long header prepended to the

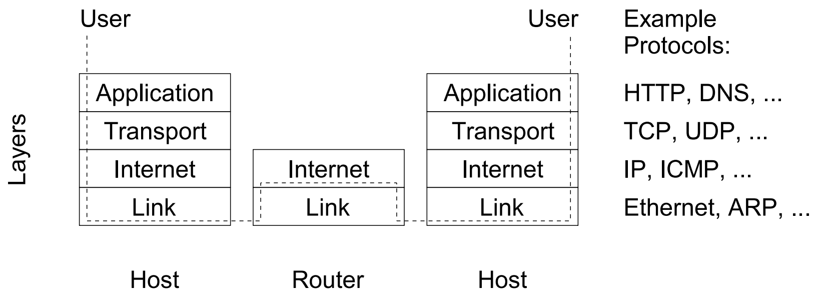


Figure 2.10: The Internet Protocol Layers

sequence of data bytes, which is in most cases a TCP segment. These segments also consist of a header of 20 to 60 bytes prepended to the (TCP) data. The IP header contains among other fields the 4-byte source and destination IP addresses and a checksum covering the header. The TCP header contains, for example, the source and destination ports and a checksum calculated over the TCP header, TCP data, and additional information (the source and destination IP addresses and other fields).

In 1979 the International Organization for Standardization (ISO) adopted a standard reference model for the interconnection of heterogeneous communication networks—the Open Systems Interconnection (OSI) reference model [Zim80, Sta98]. The model introduces a stack of seven layers each providing services to the upper layer and using services from the lower layer. In contrast to the Internet Protocol layers, the link layer is subdivided into two layers (physical and data link layer) and the application layer is subdivided into three layers (session, presentation, and application layer) (Figure 2.11) [Ros90, Fur03].

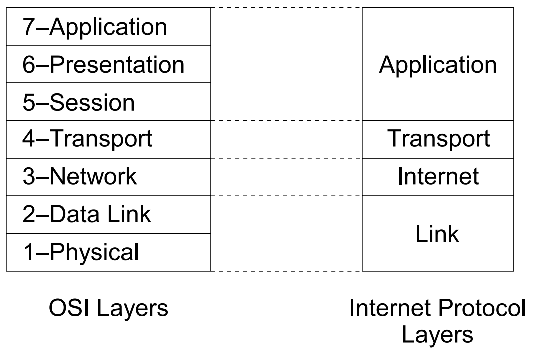


Figure 2.11: OSI Layers versus Internet Protocol Layers (following [Fur03])

At the physical layer, physical connections are established, maintained, and released (e.g., cabling, transmission of raw bits over an Ethernet). The data link layer is responsible for error detection (e.g., Ethernet CRCs). The network layer hides forwarding and routing from higher layers. The transport layer provides end-to-end error recovery and flow control. The session layer establishes, manages, and terminates connections between cooperating applications. The presentation layer resolves data representation (syntax) mismatches. The application layer comprises the remaining protocols employed between the application process and the presentation layer.

2.5 Embedded Internet

The use of standard Internet communication protocols in embedded systems is discussed extensively in the literature. Embedded Internet is used or envisioned in various application areas.

2.5.1 Application Areas

According to [WD03] the aircraft industry is moving towards Internet-based protocols. An important standard in this area is the ARINC 664 standard [ARI]. [TAP06] proposes the use of an airplane's Internet connection even for the transmission of cockpit flight data recorder data and remote control of the flight.

Internet protocols for automotive applications have been proposed for communication with car-external entities or for intra-vehicle communication. The connection to the Internet can be used for retrieving the address of the nearest gas station or restaurant. Employing standard Internet protocols on top of existing intra-vehicle bus protocols increases interoperability [JSJF98, CHB02, MB03].

Standard Internet technologies and the Internet are increasingly used for interconnecting industrial automation systems and connecting these systems to office computer systems [DNvHC05, CB05].

2.6 Gateways

When connecting differing systems, property mismatches can occur [GIJ$^+$03]. The categories of physical, syntactic, flow control, protocol, data representation, temporal, dependability, and semantic mismatches can be identified.

Gateways [Dod01, vBMM90] can be employed in order to resolve these property mismatches. A gateway is a network participant providing automated interfaces to another system using different message formats and/or communication protocols. Gateways may include forwarding, translation, and cross-banding functions.

Forwarding is the process of selectively redirecting information from one system to another without modifying the format of the data. Translation is the process of converting one data format to another while preserving as much information as possible. Cross-banding is the process of redirecting information from one communications medium to another.

[vBMM90] introduces two approaches for concatenating two differing communication protocols: service adaptation and protocol conversion. Service adaptation is the approach of employing the existing service interfaces of the two protocols to be concatenated (see Figure 2.12). Protocol conversion uses existing service interfaces of the lower protocol layer for the concatenation of the two protocols (see Figure 2.13). The advantage of service adaptation is that it can more easily be implemented. The advantage of protocol conversion is that a wider variety of optimizations can be implemented due to the direct access to low-level service interfaces.

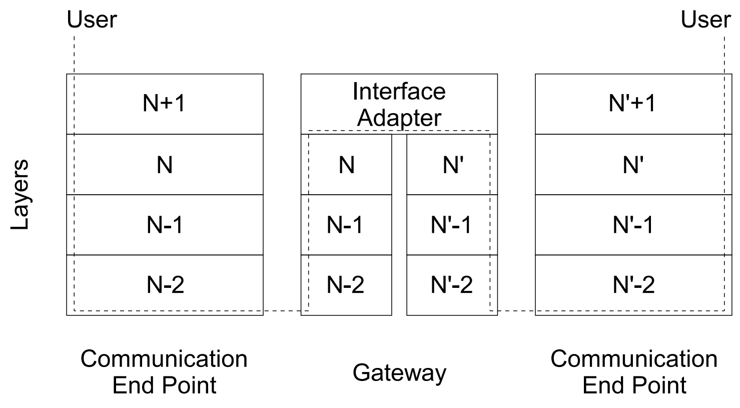


Figure 2.12: Service Adaptation (following [vBMM90])

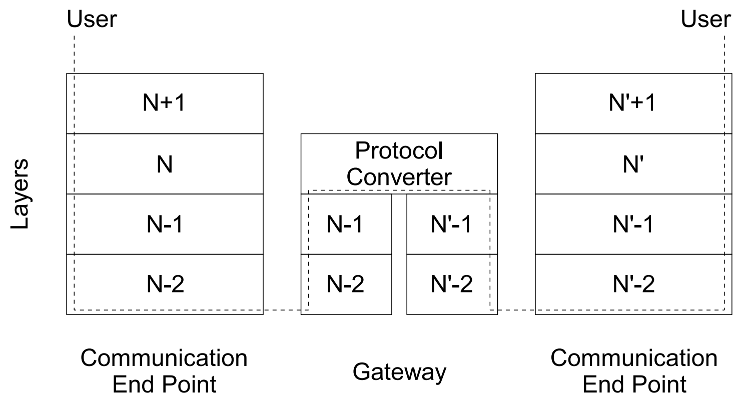


Figure 2.13: Protocol Conversion (following [vBMM90])

2.7 Security

Computer systems need protection against malicious activity: computer viruses [SL01] and Denial-of-Service (DoS) attacks [Gar00, Pis02] are prominent examples of security threats.

[AAC$^+$03] presents fundamental concepts of security based on existing concepts of dependability (see section 2.1.1). Security is the concurrent existence of availability for authorized users only, confidentiality, and integrity in the sense of absence of unauthorized system state alterations.

2.7.1 Security Goals and Rules

Security goals specify high-level security properties expected from a system. The specification of security goals enables to decide if a system exhibits a security failure [AAC$^+$03]. For example, if confidentiality of a data set is not specified as a security goal, the unlimited disclosure of the data set is no security failure. Examples of security goals are: “only administrators shall change server A’s configuration” or “only user group B shall read data set C”. A violation of such a goal is a security failure.

As with other system services, each user group might expect other security services from a system. For example, the end user of a cell phone might expect confidentiality of conversations while the cellular service provider might expect the system to allow only paying customers to use the cellular service. As a consequence, the security goals often have to merge various expectations [RRKH04, KLM$^+$04, AHS06].

In contrast to security goals, security rules are low-level constraints on the socio-technical system aiming to guarantee the fulfillment of the security goals [AAC$^+$03]. For example, the rule “administrators must keep their passwords secret” might help to fulfill the goal “only administrators shall change server A’s configuration”. A violation of a security rule is not necessarily a security failure.

The union of security goals and security rules is defined as the security policy of the system [AAC$^+$03].

2.7.2 Fault Model

[AAC$^+$03] presents a general security fault model based on concepts of dependability. A security failure is the violation of a security goal. A security error is an internal detectable security impairment. A security fault is the adjudged or hypothesized (see section 2.1.1) cause of a security error. The three main types of security faults are attacks, vulnerabilities, and intrusions. An intrusion is a malicious software-domain operational fault originating external to the system. The term operational indicates that it is created while the system is operating (in contrast to faults created during system development). An attack is an intrusion attempt and a vulnerability is an internal security fault that might lead to an (at least partially) successful attack. The relation of attack, vulnerability, and intrusion is illustrated in Figure 2.14 and an example causal chain is depicted in Figure 2.15. The interpretation of failures as faults in Figure 2.15 can be more precisely described as the perception of errors of one system (becoming visible at the interface and causing the failure) as faults from the point of view of another system (compare Figure 2.1).

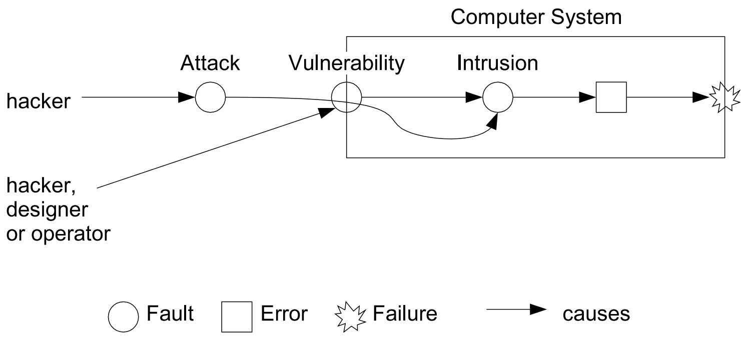


Figure 2.14: Attack, Vulnerability, and Intrusion (following [AAC+03])

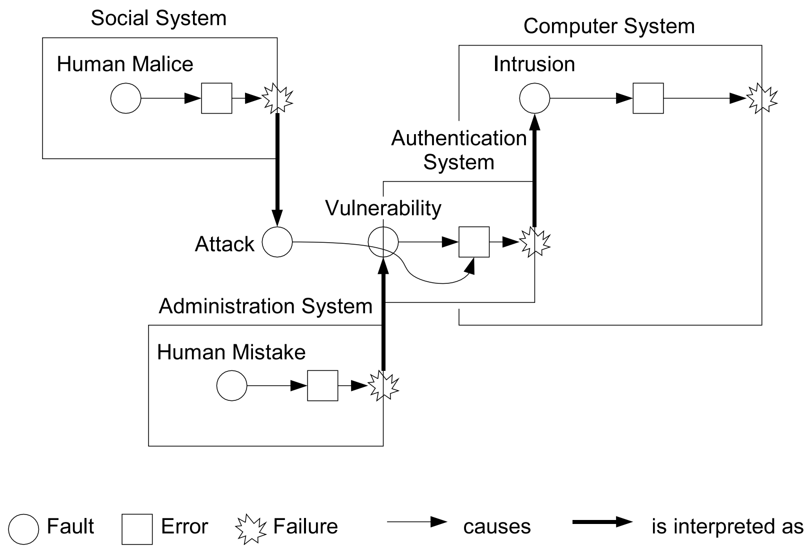


Figure 2.15: Attack, Vulnerability, and Intrusion Example (following [AAC+03])

Attacks may be subdivided in a part consisting of human activity such as planning an attack and a part consisting of technical activity, e.g., a connection attempt to the attacked system.

The importance of studying faults in order to avoid system failures is emphasized by the proposed establishment of a fault hypothesis at the beginning of the design process for fault-tolerant systems [Kop04]. In the security context, [PX05] proposes the additional consideration of security faults via misuse cases in order to drive system architecture design.

The relation and the differences between treating non-malicious and malicious faults have been discussed in the literature several times. [Rus99] compares the problem of partitioning in the avionics domain with security problems such as confidentiality. [Jon06] aims at integrating concepts from the traditional dependability domain (concerned with non-malicious faults) and the security domain (concerned with malicious faults). [Pus06] mentions a bus guardian (originally intended against non-malicious faults) as a defense against bus jamming attacks originating at a malicious node in a TTA network. [Mea94] proposes to adopt the approach of differentiating between classes of non-malicious faults in the traditional dependability domain (e.g., arbitrary failure of a node versus a fail-silent failure) for malicious faults in the security domain (e.g., passive versus active wiretapping). The authors of [DR01a, DR01b] observe a relation between reliability and security. [Was06] points out that some similarity and relation exists between safety and security.

2.7.3 Classification of Security Methods

[AAC$^+$03] combines the four approaches to attaining dependability (see section 2.1.1) with the four security fault categories attack (in the human sense), attack (in the technical sense), vulnerability, and intrusion. Of the resulting sixteen methods of attaining security, ten are distinguishable (Table 2.4).

In Table 2.4, intrusion tolerance implies that not only correct service but also correct security service is provided despite the presence of intrusions. The techniques of fault masking and fault handling are described in section 2.1.1.

Fault removal with respect to attacks (in the human and the technical sense) actually aims at removing the entity perpetrating the attack rather than the attack itself. Attack agents are pieces of code used by an attacker to carry out an attack.

Intrusion removal occurs only in the form of attack and vulnerability removal whenever attack agents or vulnerabilities *resulting from* intrusions are removed. In these cases, not the intrusions but rather the consequences of intrusions are

Fault / Approach	**Attack** (human sense)	**Attack** (technical sense)	**Vulnerability**	**Intrusion**
Prevention (how to prevent occurrence or introduction of. . .)	deterrence, laws, social pressure, secret service. . .	firewalls, authentication, authorization. . .	semi-formal and formal specification, rigorous design and management (user education, e.g., choice of passwords). . .	= attack & vulnerability prevention & removal
Tolerance (how to deliver correct service in the presence of. . .)	= vulnerability prevention & removal, intrusion tolerance		= attack prevention & removal, intrusion tolerance	error detection & recovery, fault masking, intrusion detection, fault handling
Removal (how to reduce number or severity of. . .)	physical countermeasures, capture of attacker	preventive & corrective maintenance aimed at removal of attack agents	1. formal proof, model-checking, inspection, test. . . 2. preventive & corrective maintenance, including security patches	$\subseteq$ attack & vulnerability removal
Forecasting (how to estimate present number, future incidence, likely consequences of. . .)	intelligence gathering, threat assessment. . .	assessment of presence of latent attack agents, potential consequences of their activation	assessment of: presence of vulnerabilities, exploitation difficulty, potential consequences. . .	= vulnerability & attack forecasting

Table 2.4: Classification of Security Methods (following [AAC$^+$03])

removed. From this follows that the set of intrusion removal methods is a subset of the set of attack and vulnerability removal methods.

2.7.4 Security Technologies

In this section three important security technologies are introduced: firewalls, virtual private networks (VPNs), and intrusion detection systems (IDSs). While firewalls and VPNs aim at attack tolerance (in the technical sense) and attack prevention (in the technical sense), IDSs are meant to support intrusion tolerance, attack removal, and vulnerability removal.

Two approaches are possible to compose a secure architecture employing these and other security technologies. In the defense-in-depth approach various mechanisms are installed in a layered fashion like shells around protected subsystems. In the hard perimeter approach a single "impenetrable wall" (i.e., a security mechanism) is used to protect a system and security is not of concern inside the wall. The disadvantages of the hard perimeter approach are the unrealistic assumption of an "impenetrable wall" and the lack of protection mechanisms against attacks from inside the system [DNvHC05, WE07, Was06], [PP03, p. 467].

Firewalls

A firewall [Opp00, Zal02, PP03] is a computer system or component of a computer system protecting a computer network from malicious communication traffic originating from other networks. For example, a company's internal computer network can be protected from the Internet by a firewall (see Figure 2.16). The firewall restricts communication by allowing only certain messages to pass through it.

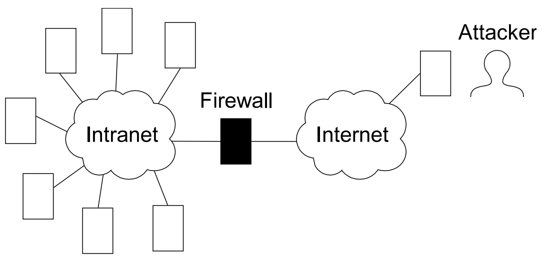


Figure 2.16: Firewall Example

Four important firewall technologies are packet filtering, stateful inspection, circuit-level gateways, and application-level gateways [Opp00].

- Packet Filtering: Packet filtering systems are restricting communication by letting through only packets fulfilling statically specified criteria (filtering rules). These criteria can contain IP address ranges and TCP/UDP port numbers but no packet data payload.

- Stateful Inspection: Stateful inspection means adapting filtering rules dynamically according to the context of a packet, i.e., packets sent or received previously. For example, packet data might be inspected in order to record whenever an application protocol requests a host on the Internet to initiate a connection to a client on the protected intranet (FTP (File Transfer Protocol) uses such a mechanism). Normally, the firewall would then dynamically allow such a connection request.

- Circuit-Level Gateways: Circuit-level gateways forward connections (multi-packet data streams, e.g., TCP connections) rather than packets. A connection from the protected network to the outside (or vice versa) is substituted by two connections: one from the protected network to the gateway and one from the gateway to the outside (or vice versa). Authentication is used in order to allow only legitimate users to connect (indirectly) to the protected network: A host wishing to establish a connection through the gateway has to gain authorization from the gateway.

- Application-Level Gateways: Application-level gateways differ from circuit-level gateways only in the fact that they operate at a higher level. The former understand application-level protocols (e.g., FTP, TELNET, or HTTP) and can thus provide more fine-grained filtering (e.g., allowing users to receive more FTP data than HTTP data).

The above firewall technologies can be combined in various ways into a firewall architecture. Two examples are screened subnet firewalls and dual-homed firewalls [Opp00, Zal02]. Figure 2.17 illustrates a screened subnet firewall. Between the Internet and the intranet an additional local network—the screened subnet—is installed. This subnet is only protected from the outside by a screening router, i.e., a router that is also a packet filter. The intranet is protected from the Internet by two screening routers and the application-level gateway. The intranet can be made visible to the outside by letting certain packets bypass the application-level gateway and pass through the two screening routers or it can be hidden by letting outer hosts communicate to it only via the application-level gateway.

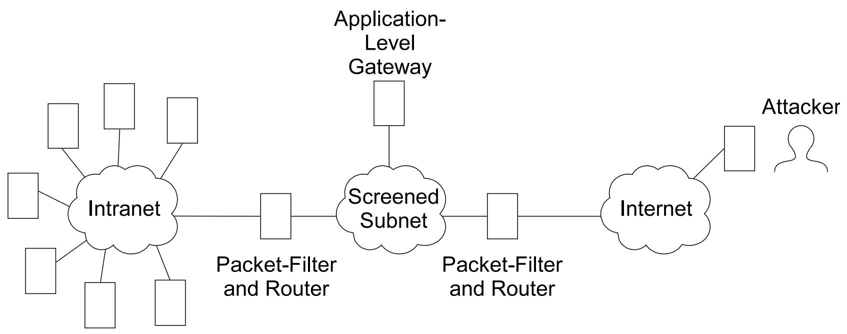


Figure 2.17: Screened Subnet Firewall (following [Opp00, Zal02])

Figure 2.18 depicts the dual-homed firewall architecture. Again two screening routers and an application-level gateway protect the intranet from the outer network. However, now the application-level gateway has two network interfaces (i.e., it is dual-homed) enabling the connection of the two screening routers. Since the application-level gateway's routing is turned off, no direct communication between the intranet and the Internet is possible. Consequently, this approach is more secure than the screened subnet architecture but due to the need for application-level gateway support for all desired application protocols it is less flexible.

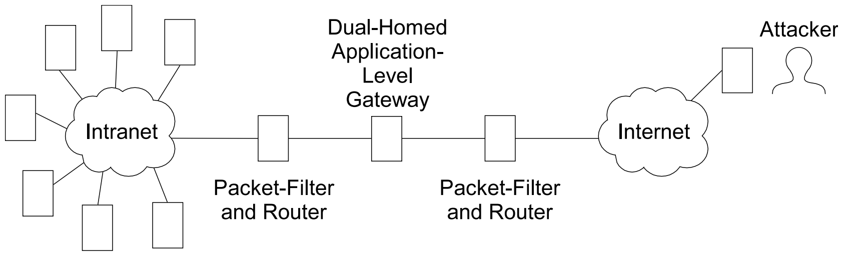


Figure 2.18: Dual-Homed Firewall (following [Opp00, Zal02])

Virtual Private Networks

A virtual private network (VPN) [Ven01, Her99, Poh01] is a communication network built on top of a communication network shared by several users (the shared network). In contrast to a real private network (e.g., a company's intranet) VPNs become private by authentication and encryption. Figure 2.19 shows an example VPN built on top of the public Internet. Two intranets (e.g., intranets of a company's two offices at different locations) are merged into one intranet by establishing a VPN tunnel between the two intranets. The data packets communicated between the two intranets are signed, encrypted, and encapsulated in IP packets routable through the public Internet. In this way, the information passing through the VPN tunnel is not visible to attackers on the Internet. However, [Opp00] points out that since normally observers of the tunnel traffic are still able to see that and when communication takes place, security is not the same as when using a real private network.

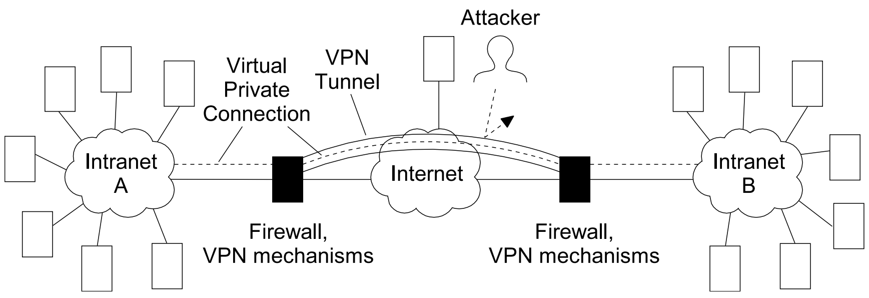


Figure 2.19: VPN Example

Generalizing from the classifications made in [CC04], VPNs can be built providing services ranging from layer 1 (L1) to layer 3 (L3) services (see the OSI reference model in section 2.4). On the other hand they can be built on a shared L1, L2, or L3 network. Prominent examples of VPN types are L3 service over L3 shared network and L2 service over L3 shared network.

Another classification focusing on usage scenarios given in [Ven01] reveals three types of VPN services: local area network (LAN) interconnect services, dial-up VPN services, and extranet VPN services.

- LAN interconnect VPN services: LAN interconnect services are used to connect two or more LANs located at various geographic areas. Figure 2.19 depicts a typical example VPN connecting two intranets. Mostly the LANs are owned by one organization, e.g., a company. For example, LAN interconnect VPN services can be implemented using L3 over L3 VPNs.

- Dial-up VPN services: Dial-up VPN services help to connect to an intranet from arbitrary remote locations. After successful authentication, the remote host is granted access to the intranet. A prominent standard for dial-up VPN applications is the Layer Two Tunneling Protocol (L2TP) [TVR+99]. It enables tunneling of PPP packets through the shared network. The Point-to-Point Protocol (PPP) [Sim94] is a communication protocol for the transport of multi-protocol datagrams over point-to-point links. For example, employees might establish a PPP connection to the company's intranet by dialing-in by modem into a modem pool located at the company. Over this PPP connection, normal IP datagrams can be transported. L2TP enables users to use a shared network, e.g., the Internet, on at least part of the communication path. In this way, long distance charges can be avoided. Two possible scenarios are client-transparent tunneling and client-aware tunneling [Cis]. In client-transparent tunneling (Figure 2.20 (a)), the remote host (e.g., the employee's PC) is separated from the tunneling end-point at the remote side, the L2TP Access Concentrator (LAC). The remote host establishes a normal PPP connection to the LAC and does not need any tunneling functionality. The tunnel and the PPP connection end at the L2TP Network Server (LNS). In client-aware tunneling (Figure 2.20 (b)), the remote host is at the same time the LAC, i.e., the remote host is aware of the tunneling. Dial-up VPN services are an example application of L2 over L3 VPNs.

- Extranet VPN services: Extranet VPN services allow non-trustworthy external hosts to access a part of an intranet. For example, a company might want to grant suppliers access to some servers inside the intranet (Figure 2.21). The accessible hosts are part of the so-called Demilitarized Zone (DMZ). The

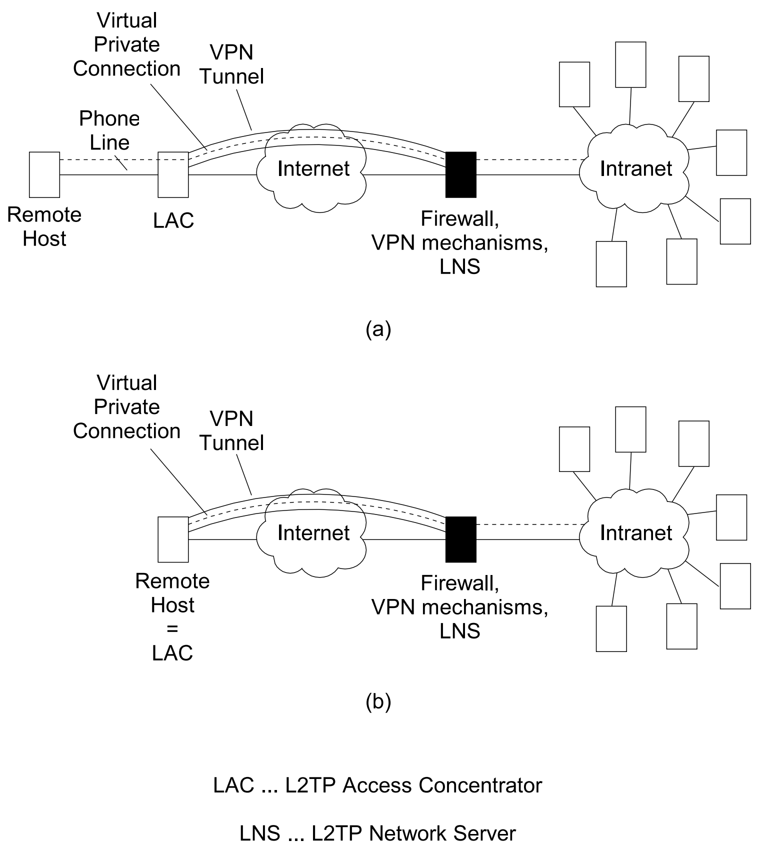

Virtual Private Connection
VPN Tunnel
Phone Line
Internet
Intranet
Remote Host
LAC
Firewall, VPN mechanisms, LNS
(a)
Virtual Private Connection
VPN Tunnel
Internet
Intranet
Remote Host = LAC
Firewall, VPN mechanisms, LNS
(b)
LAC ... L2TP Access Concentrator
LNS ... L2TP Network Server

Figure 2.20: Client-Transparent vs. Client-Aware Tunneling (following [Cis, TVR$^+$99])

non-trustworthy external hosts might connect to the DMZ by both LAN interconnect VPN services or dial-up VPN services.

Intrusion Detection and Intrusion Tolerance

An intrusion detection system (IDS) [KV02, BM01, SD02] is a part of a computer system analyzing network traffic and events in the computer system for signs of security problems [BM01].

IDSs differ in the following characteristics:

- *Information sources used for detection*—the communication network (e.g., content of packets), the host (e.g., user processes), and applications (e.g., data base queries) are the most common information sources.

- *Type of analysis performed*—the two basic categories of data analysis are anomaly detection and misuse detection.

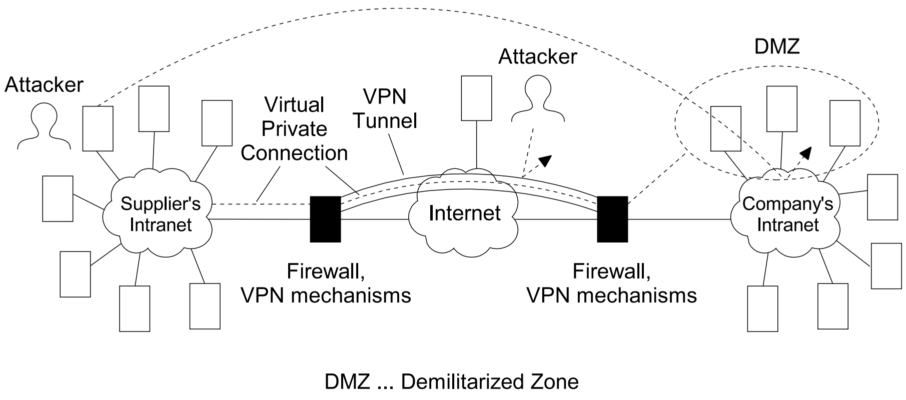


Figure 2.21: Extranet VPN Services (following [Ven01])

Anomaly detection assumes that an attack can be detected by comparing a system's monitored behavior to normal behavior. For example, an attacker who brakes into an account might use different programs at different times compared to the account's owner.

Misuse detection compares monitored behavior to typical attack and intrusion scenarios stored in a database. Such an attack description—called signature—can be a list of accessed ports and traffic patterns during a specific attack.

Anomaly detection can even detect unknown types of attacks. On the other hand, anomaly detection produces more false alarms than misuse detection due to the often unpredictable behavior of users.

- *Response to intrusions*—active responses can be the collection of additional information (e.g., logging the content of packets) or a change to the environment (e.g., firewall configuration changes, termination of a connection). Passive responses can be alarms or notifications to operators (e.g., popup windows).
- *Control strategy*—monitoring, detection, and response can be controlled from a central location (centralized control strategy) or rather at several different locations (distributed control strategy).

Intrusion tolerance [VNC03, AAC+03] aims at delivering correct system service despite the presence of intrusions. This approach recognizes the fact that fault prevention is always imperfect. Here, the term correct system service includes also security services such as confidentiality and integrity. For example, confidentiality can be guaranteed despite an intrusion if the intrusion leads to the disclosure of only a part of the confidential information and this part cannot be used to extract this information. An example intrusion tolerance method exploiting this fact is the distribution of data to different locations secured in various ways [DFF+88].

2.8 Embedded Security

Embedded security is concerned with security in embedded systems. Security is an important topic in many application areas of embedded systems such as in the automotive, aviation, railway, and industrial automation domains.

2.8.1 Embedded Security versus Office Security

Embedded system security often differs from computer security in an office environment.

Resources

Many embedded devices are low on resources such as processing power and memory. This increases the difficulty of employing standard cryptographic protocols [Koo04, KLM+04, GMF+05]. Moreover, embedded hard real-time computer systems employing cryptography have to meet deadlines in spite of time-consuming cryptographic operations [SS02, DNvHC05].

In office systems, random number generation can be accomplished by exploiting entropy in keyboard strokes and mouse movements [Ste03, ker]. Embedded systems might rather have to use hardware random number generators [HOB+06, idQ].

Physical Access

[SS02] observes that field area networks (FANs) might be located in "hostile" surroundings (e.g., remote meter reading, cars). Controlled FANs (CFANs) are defined as FANs where the physical access is controlled by the FAN owner (e.g., a normal company Intranet). Uncontrolled FANs (UFANs) are defined as FANs where physical access to the system is controlled by users who are not the owner of the FAN (e.g., a FAN for remote meter reading) and have to be considered as an adversary. Physical shielding (e.g., by seals and appropriate laws) of UFANs is emphasized as a partly solution relying on deterrence. When the embedded device is physically accessible, the power consumed or electromagnetic emanation during cryptographic computations might be analyzed in order to reveal secret information. Moreover, hardware faults might be induced in order to break the security of cryptographic operations [KLM$^+$04, Lem06]. [RPH06] proposes the use of a special hardware module—a Tamper-Proof Device (TPD)—storing keys and performing cryptographic operations in order to reduce the risk of physical intrusions.

Security Requirements

For example, in industrial automation systems, integrity and availability might have higher priorities than confidentiality [DNvHC05]. Control messages read by an adversary pose a lower risk than control messages modified or deleted by an adversary.

Consequences of Intrusions

The consequences of an intrusion into an automotive control system might be much more severe (e.g., an accident) than those of an intrusion into an office desktop computer [Koo04].

Programming Language

In the embedded system domain, often the programming language C is used. Unfortunately, in the presence of programming errors, C does not protect from buffer-overflow attacks [KLM$^+$04].

Chapter 3

Related Work

This chapter presents existing work focusing on the integration of embedded systems into the Internet and existing literature on embedded security. In the first section, interconnection architectures are examined while in the second section, security solutions are presented.

3.1 Interconnection Architectures

A possible approach to the interconnection of embedded systems and the Internet is to implement one or more TCP/IP stacks on the embedded device. However, due to limited resources of embedded systems, such a solution might be infeasible. In this case, a gateway executing a TCP/IP stack can be employed that forwards data between the embedded system and the Internet.

3.1.1 TCP/IP Stack Implemented on Embedded Device

- [CO02] presents an end-to-end remote management framework for Internet enabled devices. The aim of the work is to enable remote registration, remote configuration, dynamic updates, and device diagnostic uploads for the devices using secure communication over the public Internet. HTTP is used as interface protocol.

- The project NEXT TTA [Nex02, Nex03a, Nex03b, Nex03c] includes the specification and implementation of a generic event-triggered communication service on top of the time-triggered communication service provided by the TTA. Based on the generic ET communication service, a CAN emulation and a TCP/IP emulation for TTA nodes have been defined and implemented. The aim is to support the migration of legacy applications depending on

CAN or TCP/IP services to the TTA. On the other hand, newly developed applications can take advantage of fault-tolerance services provided by the TTA (e.g., replicated buses and bus guardians) and additional services being part of the ET communication services (e.g., a membership service for nodes participating in ET communication). For example, UDP applications can take advantage of the dependable message transmission service inside a TTA cluster and need not implement retransmission services or checksums.

3.1.2 TCP/IP Stack on External Gateway

- [IPR98] employs a gateway located at the site of the embedded system. The gateway communicates to external users by TCP/IP and HTTP and to the embedded devices by the field bus protocol. Each embedded device is represented in the gateway by an intelligent proxy object collecting data from the embedded device. Users wanting to retrieve data about an embedded device connect to the gateway which lets the proxy object retrieve the desired information from a database inside the gateway. In this way, the sporadic external requests are decoupled from the real-time control process. The proxy objects also provide additional diagnosis algorithms in order to enhance the self-diagnosis capabilities of the actual device.

- In [VP03], a gateway—the web service implementation—connected to the embedded (TTP/A) system via a special gateway node implements interfaces to the embedded system for remote access. The web service implementation uses SOAP, XML, and HTTP for the provision of these interfaces. The embedded system is protected from illegal message flows by automatic checking of message flows against a specification.

- [KKM+98] presents a TTP system that is connected to a Windows NT PC containing a TTP controller. The PC acts as a gateway between the Internet and the TTP system. TTP messages can be recorded to the PC and messages can be sent from the PC to the TTP system, e.g., the calibration of a sensor. HTML, HTTP, and remote method invocation used by Java applets are employed as interface protocols between the PC and remote clients.

- In [NFW01], two prototypes of a remote monitoring tool for railway equipment are presented. One uses HTTP, HTML, and Java applets which in turn use Java RMI (remote method invocation). Each train is equipped with a gateway—the train gateway—that acts as a RMI server and sends data about the on-board embedded system to the remote clients. Actually, an intermediary gateway—a ground station—enables the establishment of connections between remote clients and the train gateway by its connection to the Internet and its wireless connection to the train gateway. In addition, the ground station serves the HTML pages and applets in order to relieve the wireless connection from too large amounts of data.

The second prototype uses HTTP and XML/XSL as communication protocols. As in the first prototype, the train gateway connects the train's embedded system to a ground station. However, in the second prototype, the train gateway contains an HTTP server serving XML pages. The ground station is able to collect various such XML pages and to combine them in a view of all devices (or a subset of all devices) of a fleet of trains. The remote clients query the ground station for these views.

- [LGK98] presents an interconnection architecture connecting a field bus system to the Internet by an intermediate gateway locally attached to the field bus network. In the gateway, one field bus device or a collection of devices is represented in a virtual Java device. An example production cell is presented employing CORBA as communication interface between the remote clients and the gateway. As application areas, remote monitoring, remote maintenance, and customer support, but no real-time control tasks are envisaged.

- In [Dun01], at least a part of the TCP/IP stack is implemented on an external gateway. A TCP proxy is introduced in order to reduce the workload imposed on small devices. For example, the need for buffering at the small devices is reduced by reordering out-of-order TCP segments at the proxy. Furthermore, letting the proxy acknowledge data sent by the small devices reduces the time segments need to be buffered at the small devices. Moreover, the proxy stores part of the connection state in order to relieve the small devices from burden when closing connections.

3.1.3 Summary

Most existing work focuses on gateway interconnection architectures in contrast to the implementation of full interconnectivity at the communication end points. The gateway approach is motivated by limited communication bandwidth of fieldbus networks, limited resources of embedded devices, and dependability considerations. In [IPR98], it is pointed out that in an approach implementing full interconnectivity, unpredictable user requests might endanger system safety.

3.2 Embedded Security

Since embedded system security often differs from office security, frequently attention is paid to the domain of embedded security in the literature.

3.2.1 Focus on Resources

- [GMF$^+$05] presents a small SSL implementation based on elliptic curve cryptography (ECC) suitable for use on the resource constrained 8-bit Berkeley/Crossbow Mica2 “mote” platform.

- [Ste03] reports on a successful implementation of a SSL/TLS-enabled Web server application fitting into 100KB of flash memory after compression. The implementation has been done for the 16-bit 80186-based IPC@CHIP embedded Web server platform.

3.2.2 Focus on Physical Access

[SLP06] presents an overview of several possibilities of protecting against physical attacks such as power analysis attacks. For example, dummy instructions can be inserted randomly during program execution in order to complicate power analysis attacks. Another possibility is to add noise to the normal power consumption.

3.2.3 Focus on Fault Containment

In order to protect various subsystems from each other the subsystems can be implemented on distinct hardware units. However, in order to reduce hardware effort and to increase dependability (e.g., by reduction of connectors) several subsystems can be located inside one hardware component allowing for resource sharing. As a consequence, partitioning has to be employed to separate subsystems from each other inhibiting unintended interactions [OPT07, Rus99].

- [AFTO04] proposes to establish isolated (in time and space) partitions on top of a shared processor. In this so-called MILS (Multiple Independent Levels of Security) architecture, the partitioning is done by employing a partitioning real-time kernel. The envisioned application areas for the architecture are avionics and military multi-level applications. The term “multi-level” refers to multiple clearance levels.

- [Wei03] presents a security architecture for avionics. The architecture is called MLS-PCA (Multi Level Security–Polymorphic Computing Architecture) and proposes the interconnection of a large number of processes by encrypted and authenticated communication channels. Each process is hosted on its own processor and is connected to the communication network via an Encryption Process Element (EPE) dedicated to the process. The EPE can be implemented in hardware or software and cannot be bypassed by the process. Since each processor executes only a single process, the architecture implements fault containment by employing a certain physical structure instead of employing software mechanisms as it is done in [AFTO04].

3.2.4 Firewall Solutions

- [WE07] presents a security architecture for a DECOS system composed of components implemented as systems-on-chip (SoCs). A component is implemented as a single chip and consists of micro components connected to each other by a network-on-chip (NoC). The components are on their part connected to each other by a chip-external network, e.g., a Time-Triggered Ethernet (TTE) network. The chip-external network is connected to the Internet via a firewall. Furthermore, the components are also protected from the chip-external network via a firewall. Following the defense-in-depth approach, intrusion containment regions (ICRs) are established and categorized into 3 levels of increasing trust. Level 0 is the "Internet level", level 1 is the "system level" (i.e., the level of the chip-external network), and level 2 is the "chip level" (i.e., the level of the NoC). A prototype firewall is presented employing port knocking as a lightweight authentication procedure.

- [KKM+98] (see section 3.1) employs standard Windows NT security features to shield the embedded system from unwanted control signals originating in the Internet. However, in a more restrictive approach, the system can be configured in such a way that no messages at all can be sent into the cluster from the outside. In addition, due to the architectural separation of the application software from the communication controller, disruption of the time-triggered communication inside the TTA cluster by uncontrolled sending of messages from the Internet or the gateway is averted. So-called Temporal Firewall interfaces ensure that no control signal can propagate from the application software to the communication controller. A Temporal Firewall is a unidirectional data-sharing interface with state-data semantics preventing the propagation of control signals across the interface. A precise definition can be found in [KKM+98].

- [VP03] (see section 3.1) introduces a mechanism of checking message flows against a specification of correct flows. In this way, the embedded system can be protected from unwanted control actions originating from external systems.

- In [IPR98], the gateway and its contained proxy objects act also as security gateways by decoupling the sporadic external requests from the real-time control process (see section 3.1).

- In [WD03], firewalls are interposed between aircraft network types such as "avionics", "crew information", "in-flight entertainment", and "passenger".

- [Ste04] presents a security architecture for Internet Protocol-based aeronautical networks. It can be used to separate the onboard network domains "avionics", "information systems", "in-flight entertainment", and "passenger electronic devices". A firewall containing an application-level gateway (see section 2.7.4) protects networks from each other. The firewall also contains IPsec modules for the establishment of VPNs. Moreover, the firewall is a screened subnet firewall (see section 2.7.4) containing two packet filters. In order to enable access to a firewall-protected network from the outside, smart cards are proposed for storing user keys. The security model is claimed to provide a secure interface to the Internet. The model follows a defense-in-depth approach by providing various protection mechanisms such as packet filtering and cryptographic VPNs.

- In [Fur03], an example security architecture for an industrial control network is introduced. A border protecting gateway connects the industrial control network to other networks such as the Internet and the intranet. Services are accessible from the outside only through servers which are located on an isolated network segment only connected to other networks by the gateway. The gateway implements a firewall, an application gateway, and intrusion detection. Connections for the purpose of remote maintenance are protected by encryption.

3.2.5 Cryptographic Solutions

- [SS01] proposes smart cards as the preferred solution for storing and distributing keys for fieldbus/Internet interconnection. As a fall-back solution, an approach employing usernames and passwords is presented. A solution involving one-time transaction numbers (TANs) is rejected as being too cumbersome for end users. As central part of the security architecture, a gateway is employed containing a security server and an access control matrix to verify the validity of user requests.

- [NFW01] (see section 3.1) proposes the SSL (Secure Sockets Layer) protocol for Java RMI and HTTP connections and argues that SSL has become the de-facto standard over the Internet for data encryption. [CO02] (see section 3.1) also suggests the SSL protocol or alternatively the TLS (Transport Layer Security) protocol.

- [AHS06] presents a security architecture for software updates for automotive systems. Public key broadcast encryption is used for software distribution. The security requirements of various parties (e.g., embedded system manufacturer, software application developers, embedded system) are specified and an installation procedure is proposed combining several cryptographic techniques. One component of the car is a trusted component securely storing private keys and distributing received software to other components.

3.2.6 Summary

In the literature as presented in this section, embedded security is provided by efficient implementations of security protocols, physical countermeasures (physical shielding and various measures against physical attacks), virtual and physical fault containment, firewalls (packet filtering and application-level gateways), various authentication and encryption mechanisms, and intrusion detection.

Chapter 4

System Model

4.1 Overview

The system structure underlying the presented thesis is depicted in Figure 4.1. A DECOS network is connected to an Ethernet by a gateway built into a DECOS component containing an Ethernet interface. The Ethernet in turn is attached to the Internet enabling remote communication between the DECOS system and, e.g., a remote operator's Web browser executed on a PC. The gateway contains a security gateway protecting the system from attacks originating from the Internet.

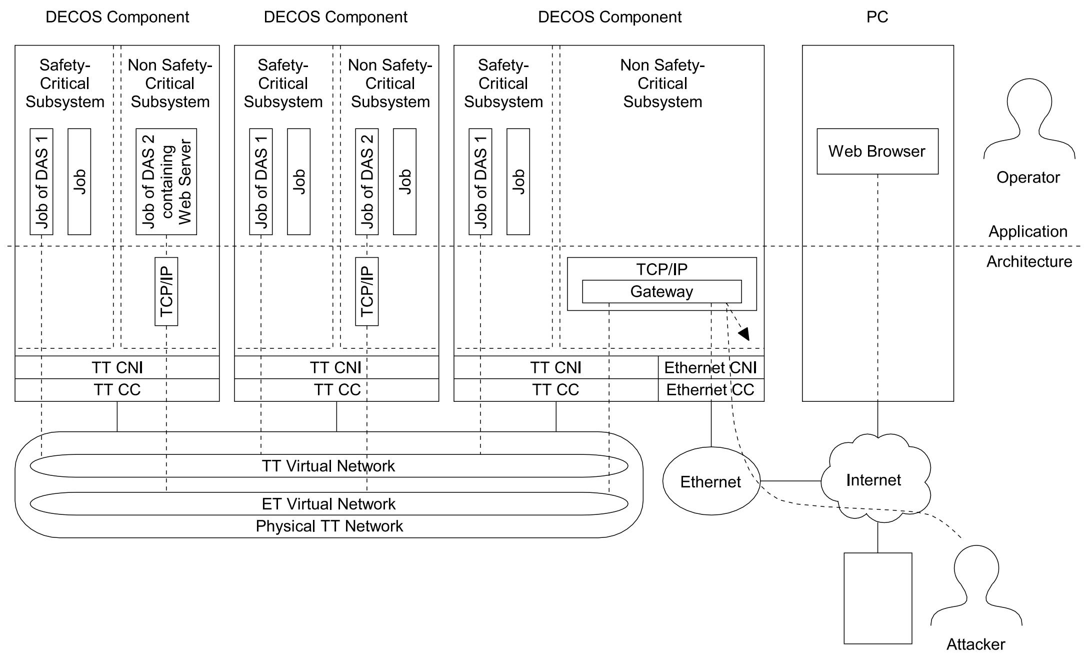


Figure 4.1: Structural Overview

4.2 Communication Services

In this section, the system's communication services are described from low-level to high-level services.

4.2.1 The Time-Triggered Physical Network

The TT physical network, e.g., implemented by a TTP/C network, supports the broadcasting of messages to all other nodes of a cluster at points in time statically defined at design-time in a cyclic schedule. Communication adheres to the TDMA scheme. While the length of a message is also statically defined, the basic connector unit must specify the content of a message at run-time. At the receiving TT CNI, the messages are not queued. Instead, an old message is replaced by the subsequent message of the next cycle. See also section 2.2.4.

4.2.2 Time-Triggered Virtual Networks

The TT physical network is used for the implementation of TT virtual networks. A TT virtual network is composed of time-triggered communication channels (TTCCs) as described in subsection *Virtual Network Service* of section 2.2.3.

4.2.3 Event-Triggered Virtual Networks

With the intention to provide a communication service for event messages (see section 2.1.3), an event-triggered (ET) communication service is built on top of the TT physical network: ET virtual networks composed of ET Communication Channels (ETCCs) (see subsection *Virtual Network Service* of section 2.2.3).

4.2.4 Ethernet Physical Network

In order to support link-level interoperability, an Ethernet interface is provided in physical gateway nodes. As ETCCs, the Ethernet service supports broadcast, dynamically configurable message lengths, dynamically configurable send instants, and queuing. Normally, Ethernets employ back pressure flow control implemented in upper layers. From this follows, that an implicit flow control scheme would lead to serious limitations of interoperability. The Ethernet service implements no error detection in the temporal domain, but CRCs for the detection of errors in the value domain.

In the next section, communication services built on top of ET virtual networks and the Ethernet service for the purpose of exchanging event messages (see section 2.1.3) are presented.

4.2.5 TCP/IP Virtual Networks

The widely used TCP/IP protocol family is employed for the establishment of TCP/IP virtual networks. Every job of a node using the TCP/IP service is considered as an IP host with its own unique public or private IP address (Figure 4.2) and its own IP send and receive queues—actually its own TCP/IP stack (the fact that the stack is implemented at the architectural level is indicated in the figure by the placement of the stack below the job). From this follows—given that every such job is equipped with its own dedicated ETCC—that varying jobs of one node can send on TCP/IP virtual networks concurrently without disturbing each other at the network level. For example, a multimedia application's UDP data stream is separated from the data stream of a navigation system due to the distribution of ETCCs to different sending slots. This is also important in the case of an erroneous sending job overloading buffers in its stack. Such a job does not disturb the other jobs on the same node. Figure 4.2 shows two DASs each with its dedicated ET virtual network. The left hidden gateway is a hidden virtual gateway connecting the two ET virtual networks. This gateway contains an IP router and possibly a security or filtering gateway limiting the set of messages allowed to traverse the gateway. The right hidden gateway is a hidden physical gateway containing an IP router and a security gateway.

Link Layer

Two kinds of networks are used as link layer networks (see Figure 4.2). The first kind of networks are event-triggered virtual networks. The second kind of networks are Ethernet networks. The hidden physical gateway contains an ARP service (see section 2.4) for the resolving of IP addresses to Ethernet addresses.

Internet Layer

An Internet Protocol (IP) service is built on top of the link layer networks. IP supports internetworking by global addressing and routing. In gateways, simple IP routing capabilities (static routing tables specified at design-time) are provided.

IP fragmentation has to be implemented and used with care since it introduces a security risk. One problem is that UDP/TCP port numbers are not contained in

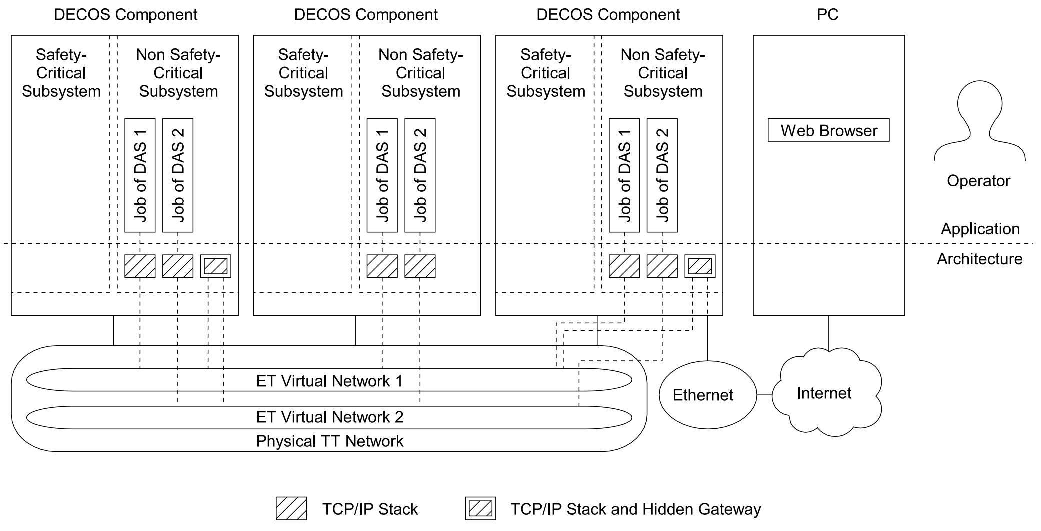


Figure 4.2: TCP/IP Network Structure

every fragment. IP fragments without information about UDP/TCP ports have to be kept in firewall buffers until the fragment containing the UDP/TCP ports arrives [KaC00]. Generally, fragments have to be kept in buffers until all fragments have arrived or a time-out has been reached. A Denial-of-Service (DoS) attack can be perpetrated by sending many IP fragments but never sending all fragments of an IP datagram. A partial solution is to assign a lower priority to fragmented IP datagrams than to other unfragmented IP datagrams. However, in this model and in the implementation fragmentation is not included.

The IP source route option inserted in the datagram header allows the sender to specify a route for the datagram. Reply IP datagrams will also be routed on this route. This allows a malicious user to pretend another identity (IP address spoofing) [KaC00, HM96]. A solution can be the employment of authentication mechanisms based on cryptography (e.g., IPsec mechanisms [Mar00]). However, in the presented model and in the implementation IP options are not considered.

IP broadcasting and IP multicasting introduces the security risk of amplified network flooding. A solution might be strong cryptographic authentication mechanisms. However, in this model and in the implementation IP broadcasting and IP multicasting is not considered any further.

The Internet Control Message Protocol (ICMP) supports IP and higher layer protocols, e.g., by reporting errors in case an IP datagram cannot reach its destination. The ICMP destination unreachable, source quench, and redirect message introduce a security risk. The destination unreachable message can be sent from a malicious host in order to impede the establishment of a connection. The source quench message might be exploited by a malicious host to cause another host to lower its sending rate and thus reduce the quality of service. The redirect message can be abused by causing a host to send its traffic to a malicious host on the local network. The ICMP echo request/echo reply (PING service) and timestamp/timestamp reply messages might be abused for the perpetration of DoS attacks by sending many requests to a host. As a limited solution, the frequency of ICMP messages accepted might be restricted, e.g., in the firewall of a cluster, in order to tolerate such kinds of DoS attacks [KaC00]. However, this enables DoS attacks on the PING service itself by sending many echo requests to the cluster. Again, authentication mechanisms such as IPsec solve many of the problems described above. In this work, ICMP is only included to an extent allowing the answering of ICMP echo requests and the sending of ICMP "Time Exceeded in Transit"-messages.

Transport Layer

On top of the IP service, UDP and TCP services are built. These offer data checksums and a host-internal address space, i.e., ports. Furthermore, TCP extends the set of services by enabling the establishment of a connection for the bi-directional transmission of ordered byte streams. The design and implementation of the TCP module is the subject of a related thesis [Ben04]. However, this TCP implementation is used for the implementation in the course of this work and the experimental evaluation.

Figure 4.3 depicts the layered structure of and message passing between the Internet protocol modules.

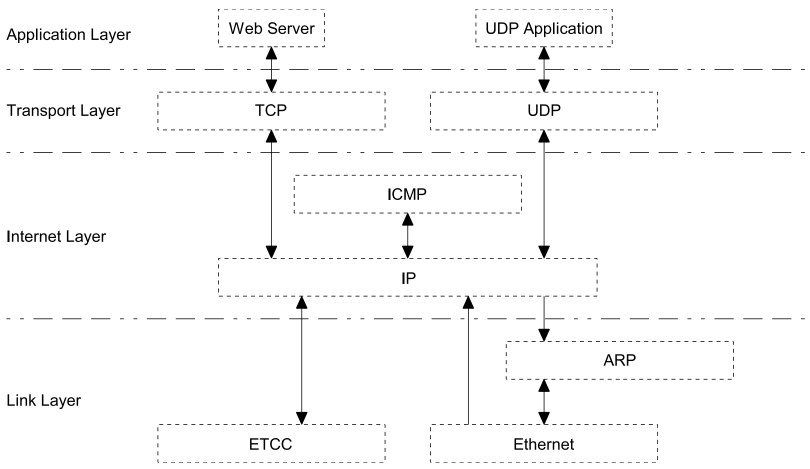


Figure 4.3: Module Overview

4.2.6 Gateways

In order to resolve property mismatches (see section 2.6) among the different communication services and protocols, gateway software and gateway hardware modules are employed in this work.

Hidden Gateways

Hidden gateways [KOPS04] are located at the architectural level which subsumes—in Internet terminology—the transport layer and the layers below. This kind of gateway is transparent to the jobs. A hidden gateway can provide IP routing and/or filtering services, i.e., the intentional dropping of messages not passing a specified filter (e.g., a firewall located in the gateway in the third component of Figure 4.1).

Hidden Virtual Gateways: A hidden virtual gateway is necessary for IP routing in the case when two or more ET virtual networks are implemented using one physical network (see Figure 4.2). In Figure 4.2 such a gateway (shaded rectangle surrounded by a rectangle in the first component) connects the ET virtual network 1 and 2.

Hidden Physical Gateways: In Figure 4.2 the hidden gateway of the third component is a hidden physical gateway resolving syntactic and protocol mismatches between the ET virtual network and the Ethernet network.

Visible Gateways

Visible gateways [KOPS04] are located at the application level and are not transparent to the jobs.

Visible Virtual Gateways: A visible virtual gateway resides at the application level and connects virtual networks. A gateway accessing both some virtual network and an IP virtual network might be a visible virtual gateway. For example, a Web server can be such a gateway: upon request, it might send data received from the virtual network out on the IP virtual network for display in a Web browser.

Visible Physical Gateways: A visible physical gateway is located at the application level and connects different physical networks. An example might be a Web server replacing the hidden physical gateway in Figure 4.2. It could serve information gathered from DASs 1 and 2.

4.3 Dependability

The system architecture aims at tolerating non-malicious hardware faults (e.g., radiation, wearout, sporadic production defects), non-malicious software faults (e.g., developmental faults), and malicious faults (e.g., unauthorized service exploitation, Denial-of-Service (DoS) attacks).

A fault-containment region (FCR) [Kop03] is a set of subsystems sharing one or more common resources. This set may be affected by a single fault. The immediate impact of certain kinds of faults is restricted to an FCR. In order to inhibit error propagation from the FCR to other parts of the system, a second independent FCR can be employed. The independent FCR must not be immediately affected by the same fault as the original FCR. An error-containment region (ECR) [Kop03, KOPS04] has to contain at least two independent FCRs in order to limit the impact of the failure of one FCR to the ECR (see *Strong Fault Isolation* in section 2.2.2).

The errors occurring in an FCR can lead to various failures of the FCR, i.e., the failure modes of the FCR. [ALRL04] identifies four viewpoints according to which failure modes can be classified: domain (timing vs. value failures), detectability (signaled vs. unsignaled failures), consistency (consistent vs. inconsistent failures), and consequences (from minor to catastrophic failures). An additional viewpoint is oftenness (permanent vs. transient failures) [Kop97].

Signaled failures occur when the failure is detected by the system and signaled. Consistent failures are perceived identically by all users of the system. A special consistent failure is a fail-silent failure, i.e., the system fails by delivering no service (in contrast to delivering some service that is incorrect). An example is a sender in a distributed system suddenly falling silent while it should continue sending messages. Minor failures have consequences of similar cost to the benefit provided by correct service delivery while catastrophic consequences have consequences of costs orders of magnitude higher than the benefit provided by correct service delivery.

An important document during the design of a fault-tolerant system is the fault hypothesis [OP06, Kop97, Kop04]. It states the assumptions about the type and frequency of faults that must be tolerated. It defines FCRs and specifies the assumed failure modes and failure frequencies of the FCRs. Furthermore, it states how long it takes to detect errors (error detection latency) and how long it takes to recover from an FCR failure (recovery interval). The recovery interval is the time from the start of the FCR failure to the point in time when the system has reached a correct system state again.

When an FCR fails an error becomes visible at its service interface. From the service consumer's perspective, this error can be interpreted as an external fault (see Figures 2.1 and 2.2). The design goal is now to tolerate this fault [OP06]. Since one fault causing an error in an FCR (e.g., EMI hitting a node) can lead to various failures of the FCR (e.g., a message is sent too late or a message has incorrect content), various mechanisms might be needed to tolerate every kind of FCR-failure (e.g., bus guardians for message-timing failures or TMR for value failures not detectable by checksums).

4.3.1 Non-Malicious Faults

Non-malicious faults are introduced by natural phenomena or humans without a malicious objective [ALRL04]. In order to illustrate the concepts (which are used for malicious faults later), also a few examples for non-malicious faults are given in the following.

Hardware Faults

With respect to many hardware faults such as radiation, electromagnetic interference (EMI), or wearout, a node is considered as an FCR. The hardware faults have to be limited to sufficiently small areas. (A contrary example are massive transient disturbances [Kop03] by EMI impacting several nodes at the same time.) The failure mode of a node is assumed to be arbitrary. The frequency of permanent hardware failures is assumed to be in the order of 100 FIT, i.e., a Mean-Time-To-Failure (MTTF) of about 1000 years. 1 FIT (failure in time) equals 1 failure in 10^9 hours, i.e., a failure rate of 10^{-9} failures per hour. The MTTF is defined as $(\text{failure rate})^{-1}$ [Kop97, Ele]. The MTTF with respect to transient hardware failures is assumed to be in the order of 100000 FIT, i.e., a MTTF of about 1 year. Only a single hardware FCR is assumed to fail, i.e., a component may fail only if no component has failed before or the system state is correct again after a previous failure [OP06, Kop04]. The DECOS fault hypothesis for non-malicious hardware faults is further detailed in [OP06].

In the following, in order to illustrate typical fault tolerance concepts later employed for malicious faults, examples are given how the union of FCRs can become ECRs. For clarity of the concepts, the first example is abstracted from the actual scenario in the DECOS architecture.

An example ECR consists of two communication links [Kop97] (Figure 4.4). In this case the ECR consists of these two FCRs. A non-malicious hardware fault contained in this ECR is EMI impacting one of the communication links. A class of errors visible at the service interface of a communication link (e.g., a detectably incorrect message (invalid checksum, incorrect message length) is delivered) can be contained inside the ECR if only one of the links fails at one time. The receiver can tolerate the failure of one link by using the correct message of the other link. Outside of the ECR, the erroneous state will not become visible (except during error detection by the receiver and for diagnostic purposes).

The maximum failure frequency that can be tolerated depends on the communication parameters. The minimum tolerated interval between two successive failures hitting different communication links has to be derived from these parameters. The error detection latency in the case of a link failure depends on the link's bandwidth, its propagation delay, and the time it takes to compute the checksum or to check the message length. The recovery interval depends on the implementation of the link. In case at some point in the middle, the link contains electronic modules, e.g., an amplifier, the modules might have to be reset. In case the link is only a cable, nothing might have to be done in order to recover from the error.

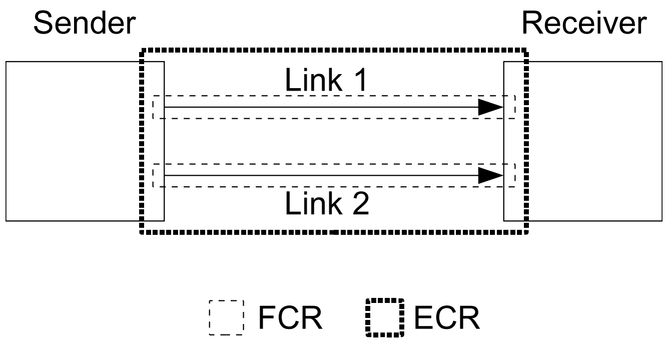


Figure 4.4: Replicated Communication Links

For example, another ECR can be established by complementing a sending node by a bus guardian [Kop03]. The sender and the bus guardian form the two FCRs of the ECR. The guardian prevents messages sent by the node outside the specified time slot from propagating to the communication network (see subsection *Strong Fault Isolation* of section 2.3.1). It is assumed that the bus guardian adheres to some restrictions such as that it cannot forge checksums, spontaneously produce correct messages, or introduce arbitrary transmission delays. If the guardian receives an untimely message it truncates the message. It is assumed that such a message can be detected as an erroneous message by correct receivers. As a consequence, the ECR can contain a timing error of a message sent by the node. The timing error does not propagate outside the ECR. However, in this case the error (the timing error) is not completely contained in the ECR but rather converted to an error in the value domain that can be detected. In order to completely contain the error in a specified subsystem, other additional mechanisms, e.g., replication (two node-guardian pairs), have to be employed. In addition to the timing error of the sender, also certain errors of the guardian can be contained in the ECR. For example, if the guardian allows all messages to pass through, this error does not propagate outside the ECR. (The FCRs are assumed to fail independently, i.e., in such a case the sender is assumed to send a valid and timely message.) Another case is a guardian error truncating a valid and timely message. Also in this case, the error does not propagate outside the ECR because it is assumed that the message is detected as an erroneous message by correct receivers.

Adapting examples in [Kop03, Kop97], another type of ECR can be identified in the case of triple modular redundancy (TMR). Three replicated jobs produce the same messages and send these messages to three replicated receiver jobs (each receiver job receives three messages). Each of the three receiver jobs performs a majority vote on the three received messages. Exact voting [Kop97, p. 133] is employed in order to determine the correct message, i.e., correct replicated messages contain exactly the same data (bit-by-bit). The six jobs are distributed to six distinct nodes. In case one of the nodes containing a sending job is hit by a hardware fault leading to the sending of an erroneous message in the value domain, that even cannot be detected by checksums or similar simple mechanisms, no error (w.r.t. the TMR sending jobs) will propagate outside the ECR (except the time until error detection at the receiver jobs and for diagnostic purposes). In this case, the ECR consists of the three sending nodes. A kind of error that can be contained inside such an ECR is a single message value error of one of the sending jobs while all three receiver jobs receive at least two correct messages.

Software Faults

With respect to non-malicious software faults (e.g., developmental faults such as bugs in the code or faults caused by specification errors), a job is considered as an FCR. If the job is replicated (probably on distinct nodes), the FCR consists of all the replicas of the job, since the jobs are assumed not to fail independently [OP06]. Below, in these both cases this FCR is referred to as the “job FCR”.

For instance, for a memory write access violation bug in a “job FCR” the ECR consists of the “job FCR” and the job partitioning mechanisms (e.g., operating system and hardware memory partitioning) forming a distinct FCR. The partitioning mechanisms have built in sufficient redundancy (legal memory ranges for the jobs) in order to compensate for the fault in the “job FCR”. The memory access violation error (the executed instruction to write to an illegal address) is contained in the ECR by aborting the execution of the instruction. Actually, the error probably cannot be contained in the ECR completely, because the missing correct write instruction might lead to other errors propagating to other components. However, the error is converted to a detectable error. In order to completely contain such an error in a specified subsystem, software design diversity [ALRL04] might be employed.

4.3.2 Malicious Faults

Malicious faults are introduced by humans with the malicious objective of causing harm [ALRL04]. With respect to malicious faults, FCRs and ECRs can be varying subsystems as outlined in the following. At first, system external malicious faults and then system internal malicious faults are considered in the following sections:

- System External Faults:
  - Direct Physical Access Attacks
  - External Unauthorized Service Exploitation Attacks
  - External Service Masquerade Attacks
  - External Eavesdropping Attacks
  - External Denial-of-Service Attacks
- System Internal Faults:
  - Node Internal Faults (Intrusions) at Subsystem-Granularity
  - Node Internal Faults (Intrusions) at Job-Granularity

The above faults are general faults threatening the general security goals "availability for authorized users only", "confidentiality", and "integrity in the sense of absence of unauthorized system state alterations". These general security goals are assumed for the system model in order to enable a general system model. [RRKH04] also mentions a few general security goals similar to these goals.

[AAC$^+$03, pp. 22, 26] mentions the possibility of also considering states of the adversary in the model. In various cases of malicious faults, this is done in the following.

Direct Physical Access Attacks

Many embedded computing devices provide an interface for reading and writing to and from memory. This interface allows the connection of the embedded device to a workstation executing a programming application which can then upload executable code and data to the device. In many application scenarios such an interface might enable an attacker to manipulate the system. An example attack is the unauthorized modification of software in order to increase the engine power of a car to an extent putting safety at risk or damaging the car. The resulting harm can be blamed on the manufacturer if no security mechanisms are built into the architecture [KOPS04]. Another example is an attack analyzing the power consumed by an electronic device in order to reveal a secret key stored inside the device. These kind of attacks are called power analysis attacks [SLP06].

One possibility of restricting access to the device is physical shielding. This can be accomplished by locking the nodes in sealed boxes. Appropriate laws prohibiting the breaking of the seal often prove to be a strong deterrent [SS02].

Another possibility is the implementation of a node as a system-on-chip (SoC) as proposed in [KOPS04]. Smart cards are examples of such systems providing restricted physical access by design. Microprocessor and memory are implemented in a single chip providing only a few pins as external interface. For example, smart cards can be programmed by writing the code into ROM at the mask-producing stage [NM96]. (However, even smart cards can be successfully intruded, e.g., by power analysis [MDS02, SLP06].)

In the case of sealed boxes or SoCs, direct access to the hardware is impeded and the remaining interfaces can be secured by authentication and other security mechanisms. For example, the only connection of a node to the outside world can be a few pins or cables leaving the sealed box in a secure manner (e.g., the hole in a sealed box through which the cable leaves the box must pose a sufficiently low security risk).

In both cases presented above, right before the system is passed to the end-user, a node can be seen as a box physically shielded against unauthorized access and offering only a number of communication interfaces to the environment (Figure 4.5). A subset of attacks on the node without using these communication interfaces is assumed to be successful only with a sufficiently low probability. The FCRs with respect to such an attack are the attacker and the physical protection mechanisms. The ECR consists of these two FCRs. The errors contained in the ECR are the states of the attacker trying to connect the node to a programming device, analysis device, etc. The errors are "detected" by the physical protection and contained in the ECR, i.e., they cannot propagate into the node. A kind of error possibly not contained in the ECR and propagating outside the ECR and into the node can be observed during the physical destruction of the node, e.g., by exposing the node to high temperatures.

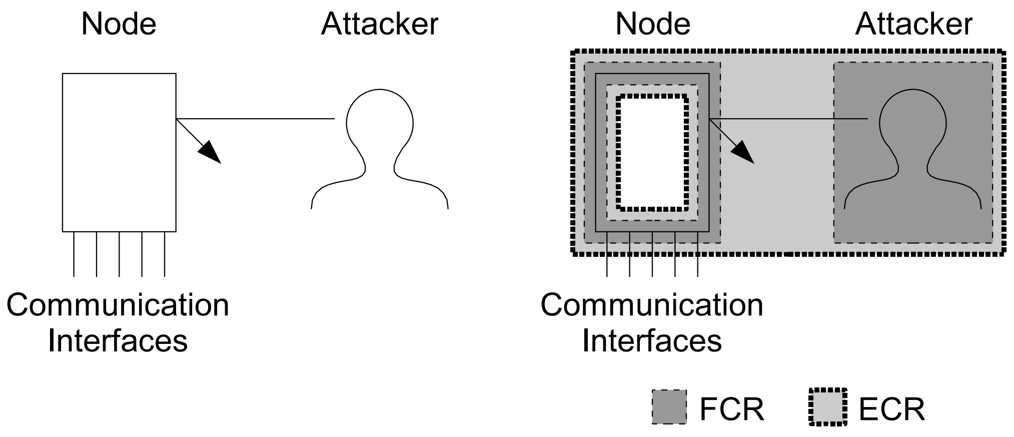


Figure 4.5: Physical Access Model

The following establishment of FCRs and ECRs does not consider any direct physical access attacks as done in the physical access model illustrated above. In contrast, the attacks examined aim at exploiting the non hard real-time TCP/IP communication interfaces of the provided communication interfaces only. For example, attacks exploiting hard real-time communication between jobs of the safety-critical subsystem are not considered.

External Unauthorized Service Exploitation Attacks

Jobs providing services via TCP/IP virtual networks connected to a local network or the Internet can be targets of service exploitation attacks. As a consequence, protection mechanisms are desirable allowing only authorized users to access the services offered. An example of unauthorized service exploitation is the unauthorized upload of job software. At the architectural level, two authentication modules, one in the DECOS node and one in the legitimate external host can be installed (Figure 4.6). One of the two FCRs is the attacking host. The second FCR is the authentication module in the DECOS component. As a consequence, the ECR consisting of the two FCRs can contain the error of the attacking host trying to exploit a service provided by the job. For example, authentication can be performed by employing asymmetric encryption using two private/public key pairs (one pair for the job and one for the legitimate host).

The maximum number of attacks tolerated depends on the strength of the authentication mechanisms used. The number of allowed service accesses per unit of time can be restricted to a maximum in order to reduce the risk of a successful brute force attack (many keys are tried). The independence of the FCRs has to be established by limiting the immediate impact of an attack (in the human sense) to the attacking host. For example, the attacker must not have access to the authentication modules.

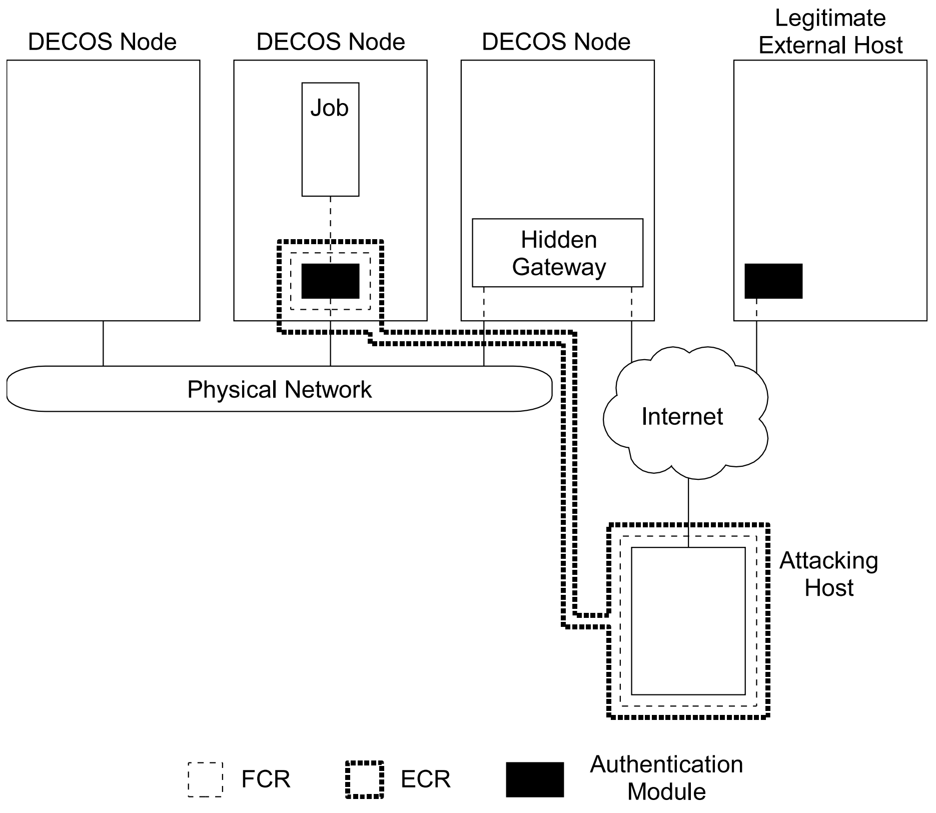


Figure 4.6: Unauthorized Service Exploitation (Attack Tolerance)

External Service Masquerade Attacks

In order to prevent service masquerade [Bak02] attacks, authentication modules can be employed as in the scenario discussed above. Generally, a masquerade attack is perpetrated by an entity pretending to be a different entity [Sta98]. In this specific case (Figure 4.7), the attacking host pretends to be a service providing job of the embedded system. As a consequence, the legitimate host might be misled to be consuming the service offered by the job. The two FCRs of the ECR are the attacking host and the legitimate host's authentication module. The error contained in the ECR is a service masquerade error originating at the attacking host. For example, before the attack the attacking host might instruct a router to redirect traffic between the job and the legitimate host through the attacking host. In case, the legitimate host requests service from the job, the attacking host can send a forged response message misleading the legitimate host. Such a message is detected by the authentication module of the legitimate host and rejected. As a consequence, the error is contained in the ECR in the sense that the error is detected and converted to a Denial-of-Service error.

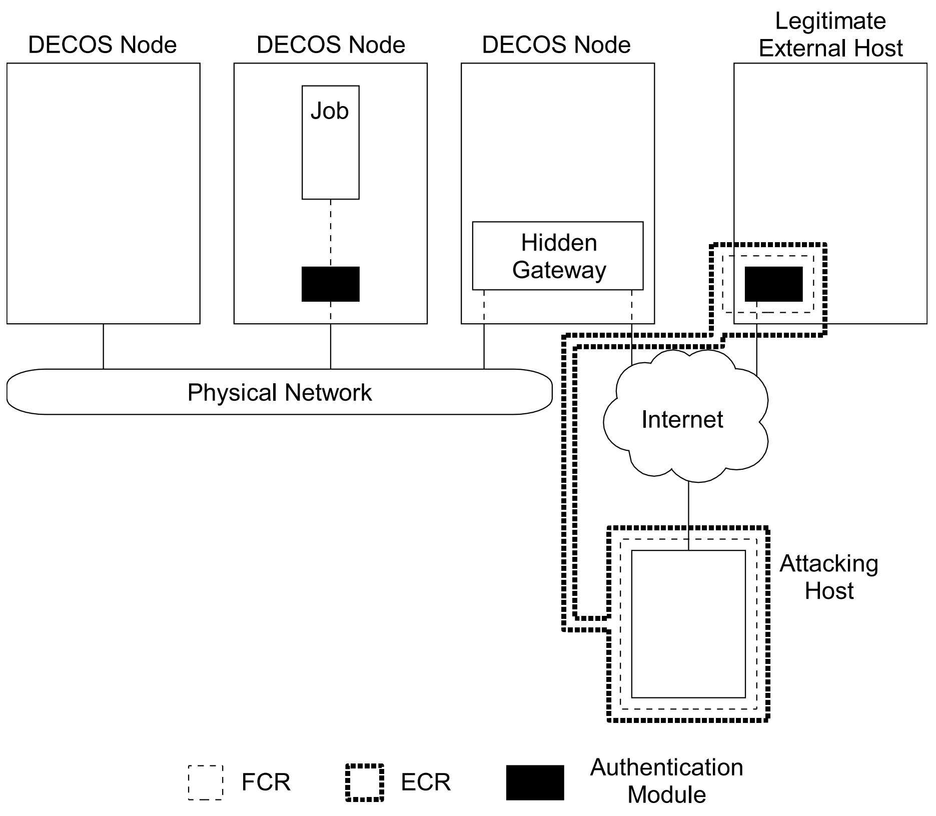


Figure 4.7: Service Masquerade (Attack Tolerance)

External Eavesdropping Attacks

External eavesdropping attacks on communication between a legitimate host and a job of the embedded system can be inhibited by encryption modules as depicted in Figure 4.8. Examples are protection of maintenance data (e.g., error statistics and information about errors that occurred during system operation) and operational data (e.g., log of the navigation system of a car). As in the case of authentication discussed above, asymmetric encryption can be employed in order to let only authorized entities decrypt the confidential data. For example, the TLS (Transport Layer Security) protocol [DA99] uses a combination of asymmetric and symmetric encryption. Symmetric keys are used for data encryption while asymmetric encryption is used for the exchange of the symmetric keys. The two FCRs of the ECR are the attacking host and the encrypted message in transit. After the FCR of the attacking host fails by perpetrating a decryption attack on the message (decryption attempt), the resulting errors (states of the decryption process, eavesdropping attempt errors) are contained in the ECR and cannot propagate outside and into the unencrypted message. The unencrypted message does not necessarily exist physically during the transmission. Physically, it might exist only in a distributed form (one part being the encrypted message and another part being the corresponding decryption key.

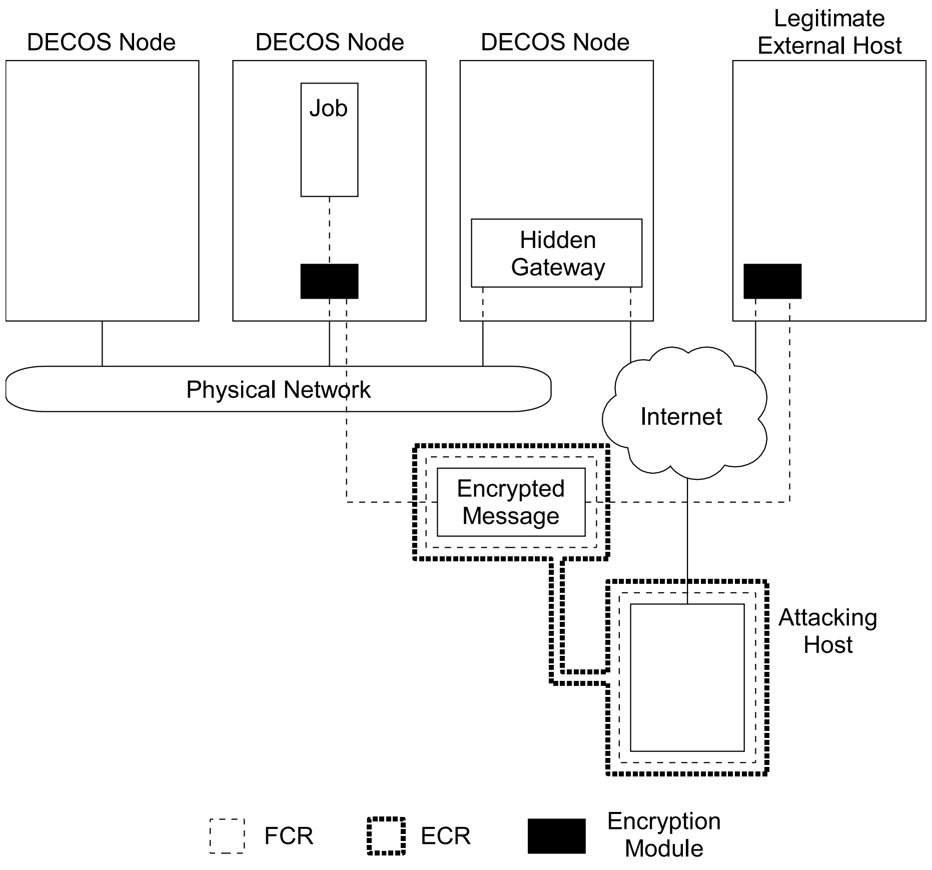


Figure 4.8: External Eavesdropping (Attack Tolerance)

External Denial-of-Service Attacks

The virtual TCP/IP networks have to be protected from Denial-of-Service (DoS) attacks [Gar00, SGR02] from external networks. An unprotected DECOS cluster could be flooded with IP datagrams originating in the Internet. A basic protection mechanism is a packet-filtering firewall (Figure 4.9). The firewall is configured to drop packets not conforming to the filtering rules. The rules contain allowed source and destination IP addresses and allowed source and destination TCP/UDP ports. The firewall and the attacking host each form an FCR. The independence of the two FCRs can be assumed when presuming that the firewall cannot be intruded from the external network. The union of these two FCRs is an ECR containing certain DoS errors. Even more severe than the case of a single attacking host are distributed DoS (DDoS) attacks [Gar00]: several attacking hosts flood a system with messages at the same time. In this case Figure 4.9 has to be adapted to contain these attacking hosts as additional parts of the attacking FCR in the ECR (Figure 4.10).

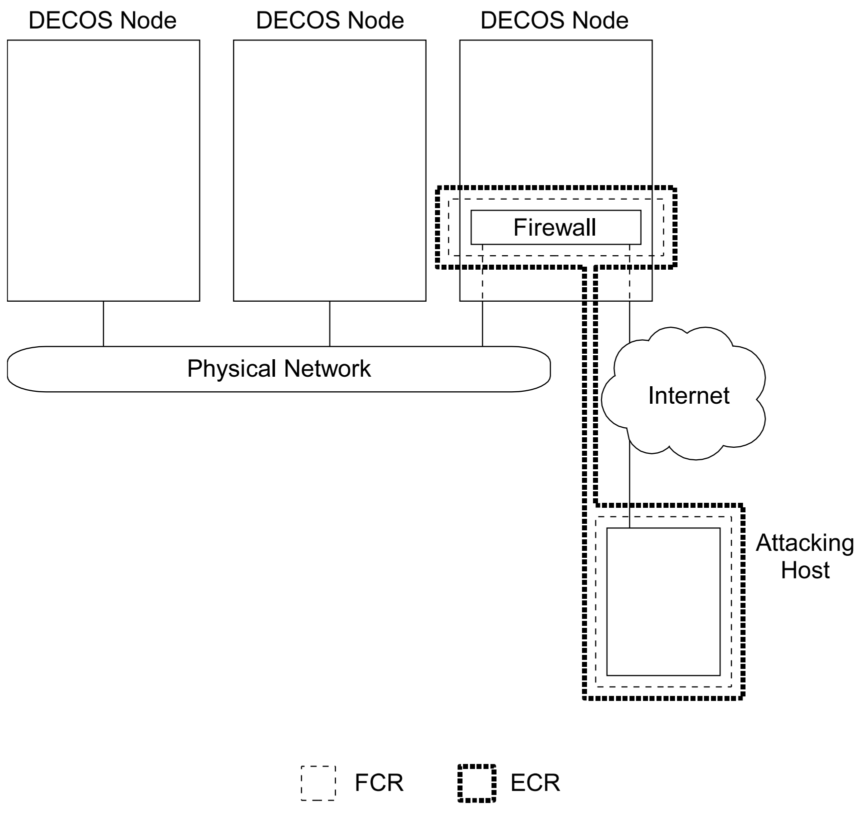


Figure 4.9: Denial-of-Service Attack Tolerance

However, a packet-filtering firewall is a solution of limited protection, since attacks can still be perpetrated by sending IP datagrams allowed by the firewall into the cluster. If the attacking host owns an IP address not allowed as source address by the firewall, it might employ IP source address spoofing [HM96] pretending to be one or more legitimate remote hosts.

As a special case, the attacking host can pretend to reside on the cluster-internal network. As a consequence, in the firewall of the cluster, ingress filters have to be employed. Ingress and egress filters check traffic entering or leaving a network for illegal IP source addresses [Gar00]. For example, the firewall of the DECOS cluster should drop IP datagrams originating from the Internet containing

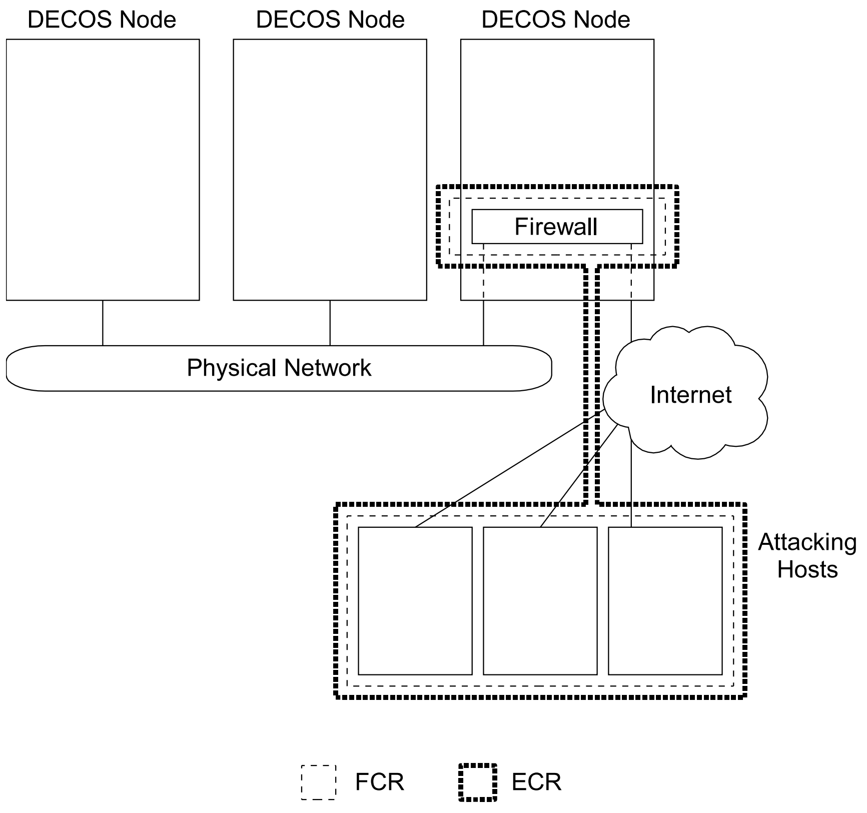


Figure 4.10: Distributed Denial-of-Service Attack Tolerance

a cluster-internal IP source address [KaC00].

Authentication protocols at the network level are a solution to the problem of IP source address spoofing. For example, the IPsec standard [Mar00] contains the Authentication Header (AH) protocol which allows the authentication of every single IP datagram. AH in tunnel mode can be used to form virtual private networks (VPNs). In this case, since no encryption is used, the word “private” has to be interpreted in a restricted sense, i.e., it is not possible for external entities to send traffic to the network but it is possible to eavesdrop on communication. Example networks are depicted in Figure 2.19 and Figure 4.11. Authentication end points in Figure 4.11 are the firewall of the car network, the car manufacturer’s firewall, the car owner’s home PC, and the car owner’s cell phone. AH in tunnel mode encapsulates an IP datagram to be sent in another IP datagram as outlined in Figure 4.12. An authentication header (AH in Figure 4.12) containing an authentication value, e.g., a hash value of the datagram parameterized by a key, is prepended to the original IP datagram. The authentication is performed for the whole new datagram excluding fields of the new (i.e., outer) header which cannot be predicted for the point of time of reception such as the TTL or the header checksum field. These and other fields might be changed at intermediate gateways [Ken05].

An additional advantage of tunneling is that the embedded system's jobs can be assigned private IP addresses (i.e., addresses not routable on the Internet and reserved for private networks) eliminating the need for the reservation of many IP addresses for every embedded system. Instead only the global IP address for the firewall has to be reserved. The FCRs and the ECR in the case of IPsec complementing the firewall (Figure 4.13) are the same as in the previous case of employing a firewall only (Figures 4.9 and 4.10). However, now a wider range of DoS attacks is tolerated, e.g., also DoS attacks employing IP source address spoofing.

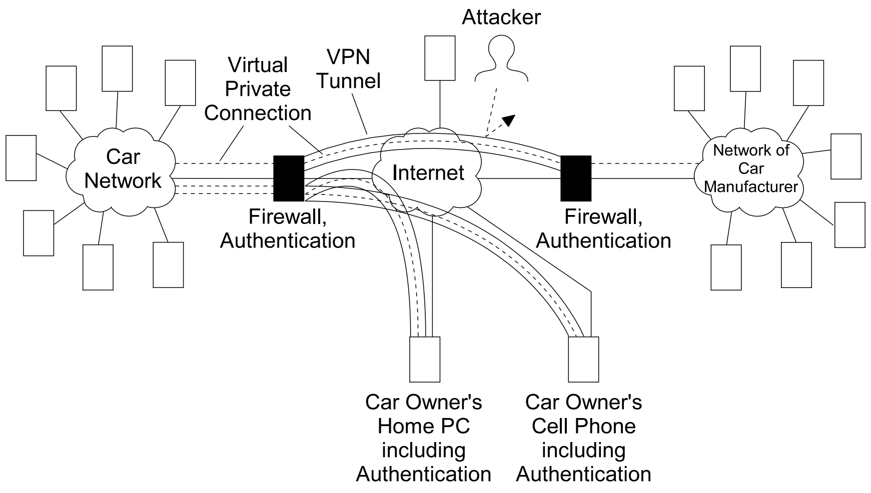

Attacker
Virtual Private Connection
VPN Tunnel
Car Network
Internet
Network of Car Manufacturer
Firewall, Authentication
Firewall, Authentication
Car Owner's Home PC including Authentication
Car Owner's Cell Phone including Authentication

Figure 4.11: Authenticated Communication Tunnels to a Car

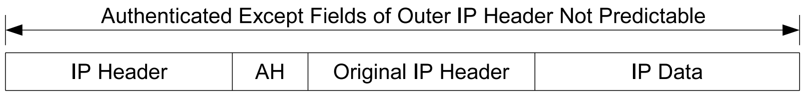

Authenticated Except Fields of Outer IP Header Not Predictable
IP Header
AH
Original IP Header
IP Data

Figure 4.12: Authenticated IP (version 4) Datagram using AH in Tunnel Mode (following [Ken05])

In order to tolerate DoS attacks, the authentication algorithms have to be sufficiently fast. Otherwise, one or more attackers can overload the firewall with arbitrary IP datagrams authenticated with a wrong key. If software algorithms prove to be too slow, sufficient speed can be reached by adding a dedicated hardware module to the embedded system's node containing the firewall [Ste04].

The problem of overloaded routers and network paths between the external host and the firewall during an attack is assumed to be solved in the Internet, e.g.,

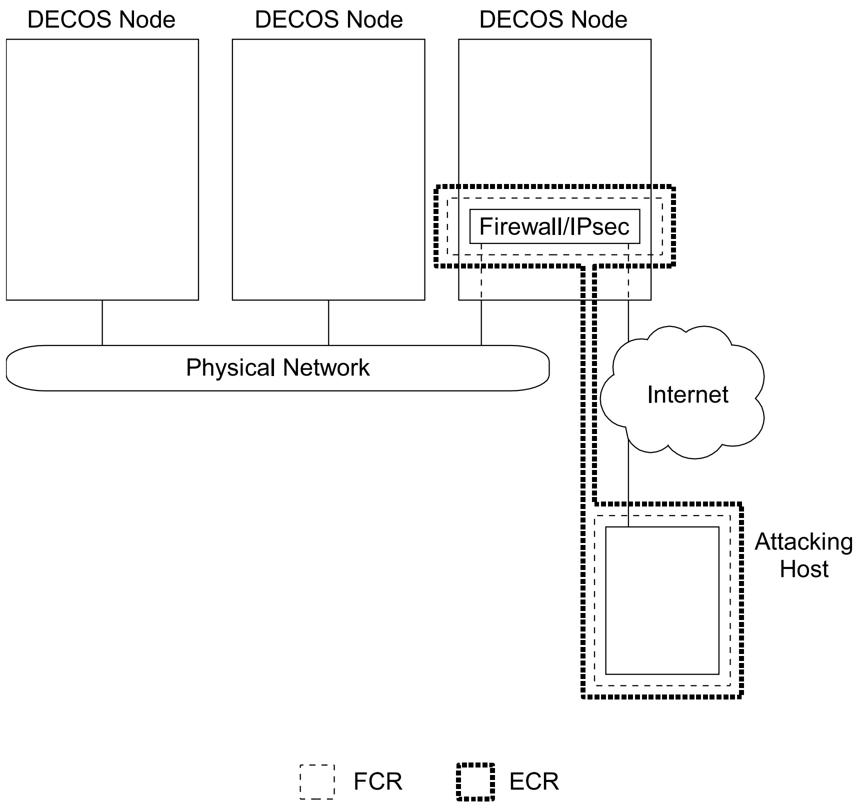


Figure 4.13: Increased Denial-of-Service Attack Tolerance

by Internet service providers (ISPs). This assumption is necessary in order to allow legitimate hosts to communicate with the cluster while an attack is perpetrated.

However, even if authentication mechanisms such as IPsec are employed, bandwidth consumption DoS attacks might still be launched from legitimate hosts, e.g., from hosts residing on the manufacturer's network (Figure 4.11). In order to tolerate such attacks or as a weaker alternative to the authentication mechanisms, an appropriate traffic scheduling discipline can be applied in the firewall. Several traffic scheduling disciplines have been proposed in the literature [Nag87, McK90, DKS89, SV96], [Nex03a, p. 44].

The aim of a traffic scheduling discipline is to allocate network and router resources (buffers, processing time) fairly to users. However, the *user* cannot easily be identified from the router's perspective since the router only knows about source and destination IP addresses and UDP/TCP ports contained in IP datagrams. A *user* could be defined as a source IP address (e.g., in the case of a user performing a file upload), as a destination IP address (e.g., file download), a source-destination pair (for balanced communication links such as telephone services), or even as a combination of source/destination IP addresses and UDP/TCP ports. As outlined in [DKS89], none of the definitions seems to be ideal. For example, defining *users* by source-destination-pairs allows malicious sources to allocate unfairly large amounts of resources by sending traffic to many destinations.

In the context of embedded systems, it seems sensible to apply a client-server model and interpret the embedded system's jobs as servers while external hosts are interpreted as clients (service users) (Figure 4.14). As a consequence, routers allocate resources fairly to external clients, i.e., depending on the direction of communication once the IP source address defines a *user* and another time the IP destination address defines a *user*. This definition is in line with the assumption that malicious hosts mostly reside outside the embedded system. Depending on the resources of the embedded system one of the approaches described in [Nag87, McK90, DKS89, SV96], [Nex03a, p. 44] seems appropriate to implement this definition of *user*. In the case of intra-cluster communication, other definitions of *users* might be employed considering application demands.

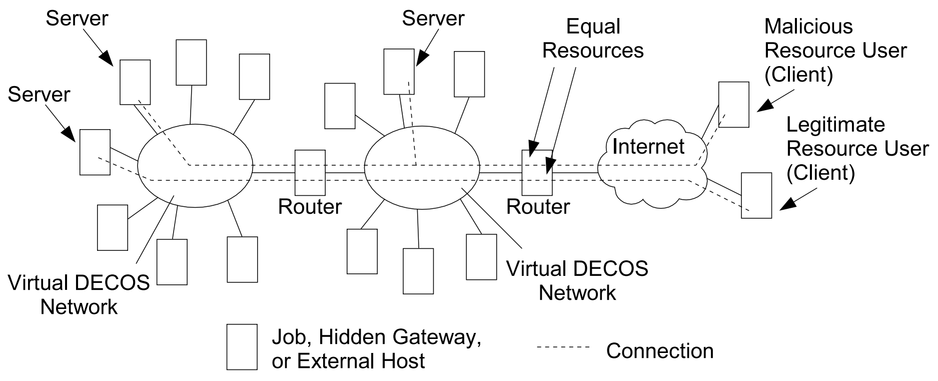


Figure 4.14: Client-Server Model

However, such traffic scheduling disciplines can only tolerate a certain number of bandwidth consumption DoS attacks of a certain severity at a time. This depends on the number of queues used, the memory available for datagram buffers, and network bandwidths. Furthermore, IP source address spoofing might still offer possibilities of circumventing the traffic scheduling mechanism if no appropriate egress filters are installed at the external network. DoS errors of sufficiently low severity and small number will be contained in the ECR and enable legitimate hosts to continue to communicate with cluster-internal jobs (Figures 4.9 and 4.10).

Node Internal Faults (Intrusions) at Subsystem-Granularity

Since the non safety-critical subsystem does not have to be certified to the highest criticality-level, an intrusion into this subsystem exploiting a hidden vulnerability might be possible with a significant probability. As a consequence of such an intrusion, an attack might be perpetrated on the safety-critical subsystem from the non safety-critical subsystem. The encapsulation service (see section 2.2.3) allocates resources such as memory and CPUs (a node should contain at least two processors) statically to the safety-critical and the non safety-critical subsystem. Furthermore, the encapsulation service does not allow information or control flow from the non safety-critical part of the system to the safety-critical part. As a consequence, an intrusion error in the non safety-critical subsystem cannot propagate to the safety-critical subsystem (Figure 4.15). An intrusion into the non safety-critical subsystem might lead to CPU occupation within the subsystem or the modification of memory content within the subsystem. Such errors are contained in the intrusion-error-containment region consisting of the respective part of the encapsulation service (first FCR) and the non safety-critical subsystem (second FCR).

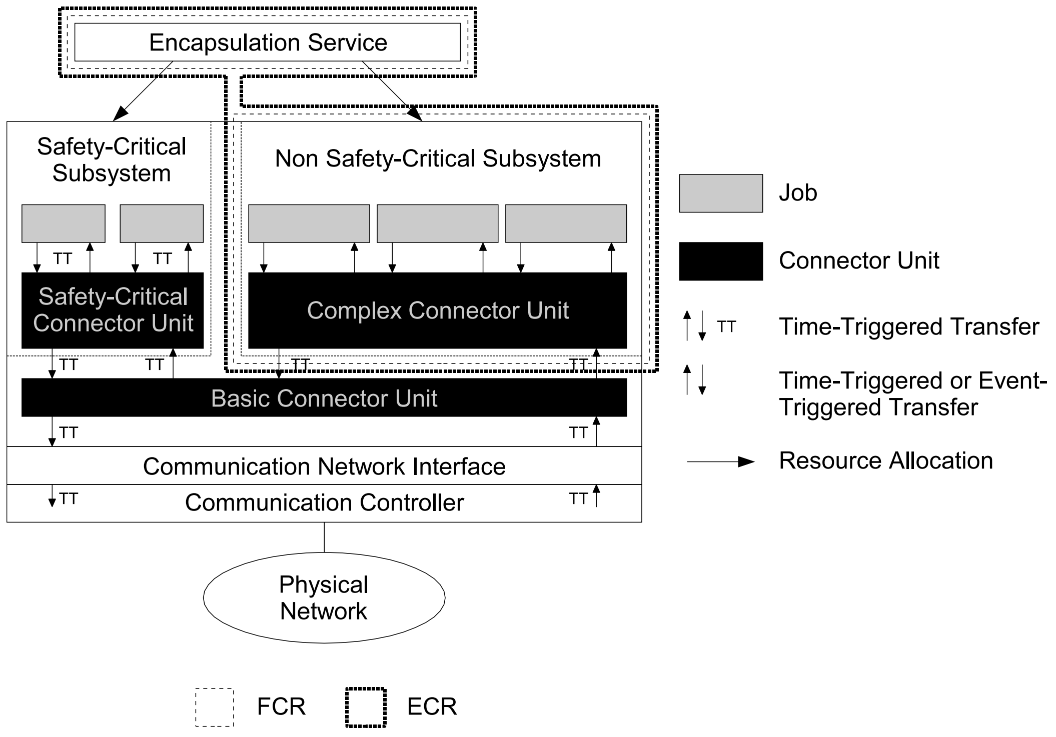


Figure 4.15: Intrusion-Error-Containment Region (Subsystem-Granularity, Component-Level) (following [KOPS04])

At the network-level (Figure 4.16), a part of the virtual network service is implemented in the basic connector unit. It statically allocates network resources to the safety-critical and the non safety-critical part of the system. An intrusion into the non safety-critical subsystem cannot compromise the communication services provided to the safety-critical subsystem. An intrusion into the basic connector unit is less probable than an intrusion into the non safety-critical subsystem since the basic connector unit has to be certified to the highest criticality level. This reduces the risk of a hidden vulnerability in the basic connector unit in contrast to the risk of such a vulnerability in the non safety-critical subsystem. The ECR consisting of the basic connector unit (first FCR) and the non safety-critical subsystem (second FCR) contains errors resulting from intrusions into the non safety-critical subsystem related to network services. For example, the content or timing of messages sent and received by the safety-critical subsystem cannot be read or modified even after such an intrusion.

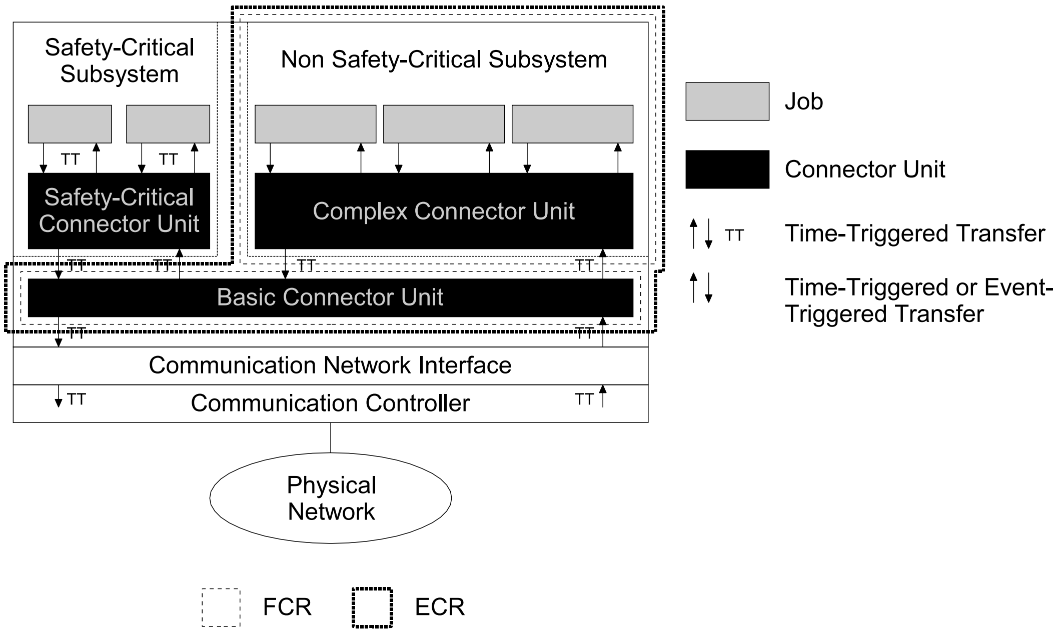

Safety-Critical Subsystem
Non Safety-Critical Subsystem
TT
Safety-Critical Connector Unit
Complex Connector Unit
Basic Connector Unit
Communication Network Interface
Communication Controller
Physical Network
Job
Connector Unit
Time-Triggered Transfer
Time-Triggered or Event-Triggered Transfer
FCR
ECR

Figure 4.16: Intrusion-Error-Containment Region (Subsystem-Granularity, Network-Level) (following [KOPS04])

Recapitulating, at subsystem-granularity, intrusion tolerance is provided at the component-level and at the network-level in the sense that the safety-critical part of the system can operate normally in the case of an intrusion into the non safety-critical part. At the network level, the separation of these two parts of the system is accomplished by the static reservation of parts of time-triggered messages. This is a similar concept as a virtual private network on the Internet. However, since confidentiality is not required in all cases, in general these networks are called virtual networks instead of virtual private networks.

Node Internal Faults (Intrusions) at Job-Granularity

In the non safety-critical part of the system, at the granularity of jobs, the encapsulation service is implemented in the operating system and possible other job partitioning mechanisms (e.g., memory range checking services implemented in hardware) (Figure 4.17). The encapsulation service protects a job from other jobs. For example, memory write operations into the memory area of another job are detected and prohibited. Such errors possibly resulting from intrusions into a job are contained in the ECR formed by the job (first FCR) and the encapsulation service (second FCR).

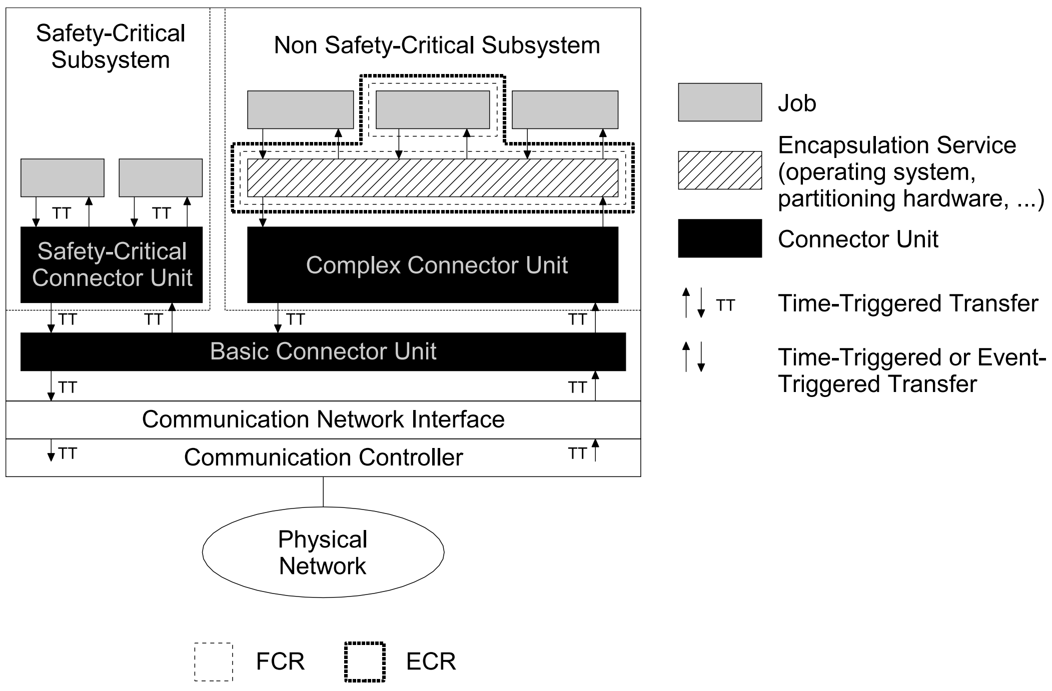

Safety-Critical Subsystem
Non Safety-Critical Subsystem
TT
Safety-Critical Connector Unit
Complex Connector Unit
Basic Connector Unit
Communication Network Interface
Communication Controller
Physical Network
Job
Encapsulation Service (operating system, partitioning hardware, ...)
Connector Unit
Time-Triggered Transfer
Time-Triggered or Event-Triggered Transfer
FCR
ECR

Figure 4.17: Intrusion-Error-Containment Region (Job-Granularity, Component-Level (1)) (following [KOPS04])

In addition to partitioning mechanisms, architectural intrusion detection services performing anomaly detection or misuse detection (see subsection *Intrusion Detection and Intrusion Tolerance* of section 2.7.4) can be employed (Figure 4.18). In this case, the FCRs are the architectural intrusion detection service and the job software module. The errors contained in the ECR consisting of these two FCRs are certain job intrusion errors. However, due to the intrusion, the error possibly cannot completely be contained in the ECR but it is rather detected and signaled to the system. The user can be informed that an intrusion has been detected and service is degraded (e.g., the multimedia system has been maliciously modified and has been deactivated). Examples of job anomaly detection at the component-level (in contrast to the network-level) are the checking of cryptographic signatures [KOPS04] of compiled job software modules and the observation of worst-case execution times (WCETs) of jobs. For example, in the case of unauthorized upload of job software, the error contained in the ECR is maliciously modified job software. The error is contained in the ECR in the sense that it is detected and can be signaled to the user.

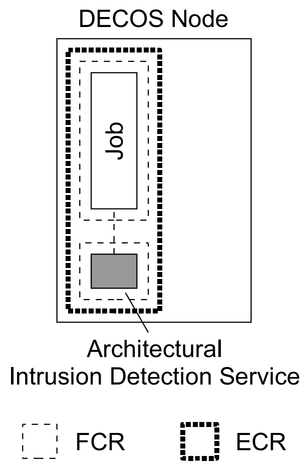


Figure 4.18: Intrusion-Error Containment (Job-Granularity, Component-Level (2))

The independence of the FCRs in Figure 4.18 is only given if the architectural intrusion detection service modules cannot be updated as easily as the job software. For example, some software modules might be allowed to be updated by the owner of the system (e.g., a car) but architectural modules might only be allowed to be updated when seals are removed, locks are unlocked, and/or the manufacturer is the authenticated entity performing the update.

The independence of various jobs with respect to intrusions due to the revealing of the private key depends on the variation of protection mechanisms. In practice, one smart card (containing the system-owner's private key) or one password might be used for all jobs which can be updated by the system-owner. In this case, all these jobs reside in one FCR (Figure 4.19). In such a case, replication cannot provide compensation and fault tolerance with respect to intrusions.

Since information or control flow from the non safety-critical subsystem to the safety-critical subsystem is prohibited, service interfaces such as software update interfaces for the safety-critical subsystem cannot be provided via Internet interfaces. However, the remote-update, configuration, etc. of the safety-critical

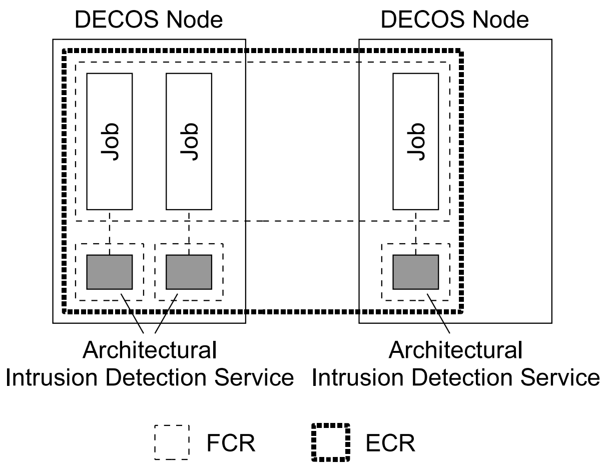

DECOS Node
DECOS Node
Job
Job
Job
Architectural Intrusion Detection Service
Architectural Intrusion Detection Service
FCR
ECR

Figure 4.19: Intrusion-Error Containment (Job-Granularity, Component-Level (3))

subsystem might be a future research topic due to the cost savings to be expected. The probability of an intrusion would have to be proven to be sufficiently low.

In the non safety-critical subsystem, again at the granularity of jobs, the network-level partitioning service, i.e., the virtual network service, is provided by the complex connector unit (Figure 4.20). A job's network resources are statically assigned to the job and cannot be accessed by another job that has been intruded. Certain intrusion errors (e.g., network overload errors and eavesdropping errors) resulting from intrusions into a job are contained in the ECR formed by the job (first FCR) and the complex connector unit (second FCR). Out-of-norm assertions (ONAs) [POK05] can be applied in the message classification layer [OPK05] in the complex connector unit. As a consequence, non-malicious and malicious faults can be detected by employing the same mechanism for both cases [KOPS04].

The employment of individual TCP/IP stacks (especially buffer space) for jobs (see Figure 4.2) is a special case of the above construction and enables the establishment of ECRs with respect to send-buffer overload attack errors (Figure 4.21). If a job has been intruded and a send-buffer overload attack is perpetrated on the TCP/IP stack, the separation of send buffers enables non-malicious jobs to send independently. Since the ET virtual networks are composed of independent sets of ETCCs, even at the network level (in contrast to TCP/IP-stack level) a malicious sending job cannot directly interfere with messages from other sending jobs on the same ET virtual network.

Finally, in [OPT07, KOPS04] it is proposed to increase partitioning even during the development phase by employing a three phase design model. First, during the requirement analysis, the system integrator (e.g., the car manufacturer) precisely

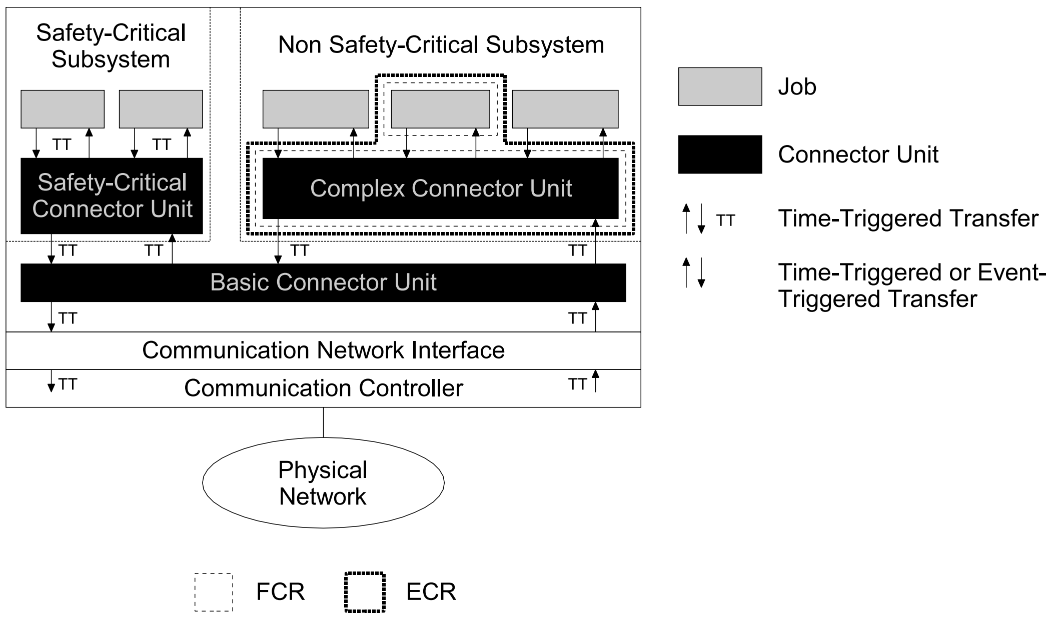

Safety-Critical Subsystem
Non Safety-Critical Subsystem
TT
Safety-Critical Connector Unit
Complex Connector Unit
Basic Connector Unit
Communication Network Interface
Communication Controller
Physical Network
Job
Connector Unit
Time-Triggered Transfer
Time-Triggered or Event-Triggered Transfer
FCR
ECR

Figure 4.20: Intrusion-Error-Containment Region (Job-Granularity, Network-Level) (following [KOPS04])

specifies the interfaces between DASs in the time and value domain. Second, during the DAS design phase, DASs are developed in parallel independently by several vendors. Finally, during the system integration phase, the system integrator composes the overall system out of the DASs. Due to the precise specification of DAS interfaces, the source code of DASs need not be revealed to the system integrator. The system integrator rather sees only compiled software modules. As a consequence, one prerequisite for the intrusions discussed above—the vulnerability—can be traced back to the DAS developer [KOPS04]. Moreover, due to the possibly higher independence of vulnerabilities an intrusion error might not easily propagate into another DAS by a second intrusion since the attacker might have to gain additional knowledge to exploit a second (different) kind of vulnerability. Possibly, DASs developed by one vendor might contain identical or more similar vulnerabilities than DASs provided by distinct vendors. In addition, the three phase design model provides an intellectual property (IP) protection service, i.e., a confidentiality service, if it is sufficiently hard to reconstruct the source code (or important parts) from the compiled code by reverse engineering.

Table 4.1 summarizes the four cases of intrusion-error containment presented above. At the component-level, the static allocation of resources to subsystems, the operating system, hardware mechanisms, and architectural intrusion detection services are employed for error containment. At the network-level, the connector

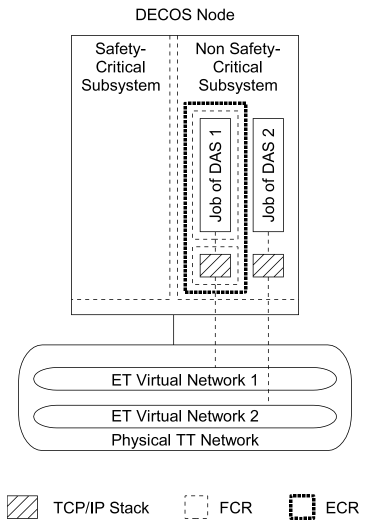


Figure 4.21: Attack Tolerance with respect to Send-Buffer Overload

units are responsible for error containment between subsystems or jobs.

	Subsystem-Granularity	Job-Granularity
Component-Level	Static Allocation of Resources (Figure 4.15)	Operating System, Hardware (Figure 4.17), Intrusion Detection (Figures 4.18 and 4.19)
Network-Level	Basic Connector Unit (Figure 4.16)	Complex Connector Unit (Figures 4.20 and 4.21)

Table 4.1: Four Cases of Intrusion-Error Containment

Overview of Fault Classes

Finally, Table 4.2 outlines which properties of security (integrity, availability, confidentiality) could be compromised by which fault class presented in the above sections. Furthermore, it illustrates which fault class is tackled by which kind of fault tolerance—attack tolerance and/or intrusion tolerance—in the model. The mechanisms introduced in the sections on node internal faults can be viewed as being both attack and intrusion tolerance mechanisms since system internal attacks might be the consequence of intrusions into the system. The last two columns indicate which kind of fault causes errors in the respective FCR in the presented model and which kind of FCR-failure can be observed as a result. In the case of FCR-failures resulting from intrusions (last column), the FCR-failure can be an attack (the second attack after the attack causing the intrusion) or another kind of failure resulting from the intrusion. For example, if the voice output of a car's navigation system fails due to an intrusion, further interpretation of this failure as an attack (e.g., on the driver) might not be necessary. Moreover, a failure being an intrusion's unintended side-effect might not be seen as a deliberate fault and therefore not as an attack. Figure 4.22 illustrates the two cases referenced by the last two columns of Table 4.2.

- ●: "fault class" compromises "security property"
- ◇: "fault class" is tackled by "fault-tolerance class" in system model
- ∗: "fault class" is fault of FCR of "which type" in system model (Figure 4.22)

Fault Class	Integrity	Availability	Confidentiality	Attack Tolerance	Intrusion Tolerance	Fault = Human Malice, FCR-Failure = Attack	Fault = Intrusion, FCR-Failure = Attack/Other Failure
Direct Physical Access Attacks	●	●	●	◇		∗	
External Unauthorized Service Exploitation Attacks	●	●	●	◇		∗	
External Service Masquerade Attacks	●		●	◇		∗	
External Eavesdropping Attacks			●	◇		∗	
External Denial-of-Service Attacks		●		◇		∗	
Node Internal Faults (Intrusions) at Subsystem-Granularity	●	●	●	◇	◇		∗
Node Internal Faults (Intrusions) at Job-Granularity	●	●	●	◇	◇		∗

Table 4.2: Relations of Fault Classes to Security Properties, Fault Tolerance Classes, and Fault Model

4.4 Applications

As outlined in section 1.1, several kinds of applications might employ the secure Internet communication services presented in the previous sections. These appli-

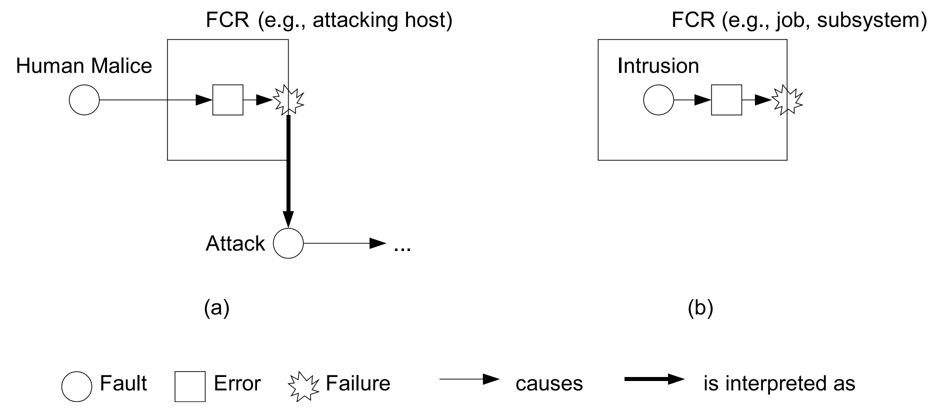


Figure 4.22: Two Kinds of Malicious Faults Leading to FCR-Failures

cations can be classified into maintenance and operational applications.

4.4.1 Maintenance

Condition-Based Maintenance

[AM04] mentions three primary maintenance strategies: Break Down (BD) Maintenance, i.e., run to failure; Planned Scheduled Maintenance (PSM); and finally Condition-Based Maintenance (CBM), i.e., maintenance decisions are based on knowledge about the current condition of components.

With CBM, condition monitoring is used in order to gather the desired knowledge during run-time. For example, wearout failures can be predicted by performing the discrete Fourier analysis of the vibrations of a train engine [IPR98]. In this way, maintenance effort can be reduced to the point when it is most probably needed. Classical approaches to maintenance such as BD maintenance or PSM might be unnecessarily costly [SLG$^+$01].

The diagnostic infrastructure introduced in [KOPS04] supports the use of condition-based maintenance (CBM) (see section 2.2.3). A diagnostic DAS analyzes gathered diagnostic data inside the embedded system. However, in case in an application area (e.g., automation or railways) performing the analysis external to the embedded system is possible and sufficient, such a solution might be preferred due to the simplification of the embedded system. In such a case, CBM maintenance can be performed by periodically collecting maintenance data via secure remote interfaces. *Secure* interfaces have to be provided in order to avoid corruption of maintenance data.

Fault Diagnosis

[KDH$^+$01] introduces an approach to fault diagnosis utilizing knowledge from many cars currently in use. By implementing appropriate remote interfaces, thousands of cars can be configured to store a specified set of internal states and send the records of such a set around the point in time of a predefined detected error. This technique enables engineers to gain deeper insight into cause-effect relationships with respect to many errors. Automated and manual data analysis can be applied to the centrally stored data.

In the integrated DECOS architecture, the sparse time base helps in taking a precise snapshot of subsystems, i.e., recording a well-defined distributed state.

Security guarantees have to be provided for the remote fault diagnosis interfaces in order to prevent corruption of the diagnostic data to be gained. Furthermore, manufacturers might not want to uncover data about system internals or the number and kind of errors detected in the field.

Engineering Feedback

For example, in the automotive domain, data from the fleet in the field can be gathered by utilizing remote interfaces to the cars. In this way, usage patterns and realistic degradation data of components can be collected in a centralized data base [KDH$^+$01]. This enables engineers to compare system models with real systems and possibly refine these models.

The authenticity and confidentiality of the data collected has to be guaranteed by appropriate security mechanisms.

Software Update and Reconfiguration

For example, in the automotive domain, high costs are involved either for users or manufacturers in the process of conventional software updates or reconfiguration activities conducted by a technician using local maintenance interfaces at the car. Software updates or reconfiguration might be necessary in the case of bug fixing, installation of new features and applications, and in the case of customization desired by the customer [CO02]. Costs can be reduced by employing secure remote interfaces for dynamic software updates and dynamic reconfiguration.

As mentioned in section 4.3.2, in this thesis it is proposed to protect the safety-critical part of the system from unauthorized modification by prohibiting any information or control flow from the non safety-critical part to the safety-critical part. As a consequence, remote dynamic software updates and dynamic reconfiguration can only be applied to the non safety-critical part. However, by following this approach the certification effort is kept to a minimum.

4.4.2 Operational

Multimedia

In the automotive domain, multimedia applications gain increasing attention. The MOST Cooperation aims to establish a common standard for multimedia networking in cars: MOST (Media Orientated Systems Transport) networks are multimedia fiber-optic networks optimized for automotive applications. Integration of various multimedia systems and integration of multimedia systems with other subsystems can be important for functional and economical reasons. Even safety might be increased by subsystem integration. For example, the stereo system should mute itself when the car telephone receives a call in order to relieve the driver of muting it. Integration of navigation and DVD playback systems by using unified displays enable passengers to switch between navigation information and entertainment easily [MOS]. An example approach of adding extensive multimedia functionality to a car is using a MOST network for multimedia applications and other types of networks (e.g., CAN networks, see subsection *Virtual Network Service* of section 2.2.3) for other applications [Emb]. However, the *integration* of multimedia systems with other subsystems reduces the weight, cost, and size of cabling. This is becoming increasingly important due to the probably further rising number of electronic units built into a car [LH02]. In an integrated DECOS system, ET communication services, e.g., TCP/IP, can be used for multimedia applications and TT services can be used for other applications and even safety-critical subsystems. In contrast to using separate buses for various application domains, an integrated DECOS architecture employs one physical network for all communication services.

In addition, many multimedia applications demand for tight integration with the Internet. This can be accomplished securely by using the architecture proposed in this thesis.

Non Safety-Critical Driver Assistance

A car's navigation system might benefit from a secure connection to the Internet for up-to-date information about traffic jams and road works. Moreover, such a connection can be exploited for the transmission of information from the car to an external traffic management center. The sending of a car's travel times helps to build up-to-date traffic models [Sch96].

Chapter 5

Implementation

5.1 The DECOS Prototype Cluster

The implementation has been done on a DECOS prototype cluster developed at the Institute of Computer Engineering at the University of Technology Vienna. The prototype cluster aims at implementing the DECOS architecture on top of a core TTA system.

The DECOS prototype cluster consists of five components communicating via a physical TTP/C network. Each component is implemented as the union of three hardware nodes (Figure 5.1). The hardware nodes implement the subcomponents of the component model depicted on the left in Figure 5.1.

5.1.1 Hardware Setup

The Basic Connector Unit (BCU), the Communication Network Interface (CNI), and the Communication Controller are implemented on TTTech [TTT] Monitoring Nodes MN222 [TTT03]. These nodes are equipped with an 80 MHz Freescale MPC855T PowerQUICC^{TM} Integrated Communications Processor and a TTP/C-C2 (AS8202) controller. In the following these nodes will be called BCU nodes. The safety-critical and the non safety-critical subsystems are implemented on Soekris Engineering [Soe] net4521 nodes each containing a 133 MHz AMD ElanSC520 (486 class) processor, 64 MB SDRAM, a 128 MB CompactFlash Card, two 10/100 Mbit Ethernet ports (RJ-45), a serial port, and two PC-Card/Cardbus slots. One of the PC-Card slots of the Soekris nodes is equipped with a D-Link Ethernet PCMCIA-Card in order to be able to transfer data to and from the development workstation.

The BCU nodes are connected to the safety-critical and non safety-critical

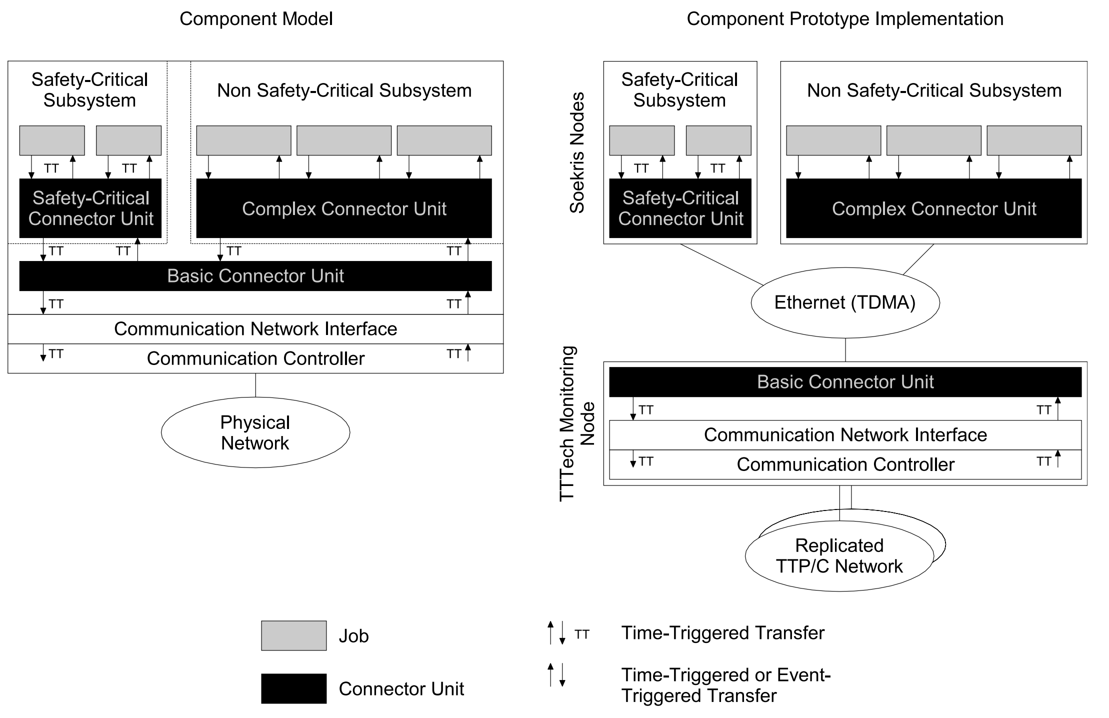


Figure 5.1: Component Model vs. Prototype Implementation

subsystems via Ethernet employing a TDMA scheme. Each such Ethernet network is implemented by an Ethernet hub. The BCU nodes are connected to each other by TTP/C. The TTP/C network is implemented by two Ethernet hubs (i.e., two replicated physical TTP/C networks are provided).

5.1.2 Software Setup

The Soekris nodes are using RTAI [RTA] version 3.1, LXRT, and Linux (kernel 2.6.9) as operating system. RTAI is a hard real-time extension to the Linux kernel. LXRT is an extension of RTAI enabling the execution of hard real-time tasks in user space. Standard Linux is executed as the idle-task in the RTAI/LXRT environment. Jobs are periodically executed as hard real-time tasks. The execution of jobs in user space allows to exploit memory protection mechanisms available for Linux user processes. The execution of jobs as hard real-time tasks while using an additionally developed scheduler enforcing job deadlines [HPOS05] inhibits temporal fault propagation between jobs and between normal Linux user programs and jobs. Hence, spatial and temporal partitioning is enforced at job level.

5.2 Virtual TCP/IP Networks

As depicted in Figure 4.2, Event-Triggered (ET) virtual networks are employed for the implementation of TCP/IP communication services. Every job owns a dedicated TCP/IP stack. The model depicted in Figure 4.2 can be mapped to the implementation as done in Figure 5.2. However, for simplicity each component is depicted as a single hardware node although the implementation uses three hardware nodes for every component as described above. In order to simplify the implementation, the TCP/IP stacks have been located inside the same LXRT tasks as the jobs. The hidden gateway on the right in Figure 4.2 is implemented by the union (big rectangle inside the right component in Figure 5.2) of a hidden gateway in a LXRT task, a hidden gateway in standard Linux user processes and in the Linux kernel, and the virtual TUN network in the Linux kernel. In this way, the existing Linux Ethernet driver can be used.

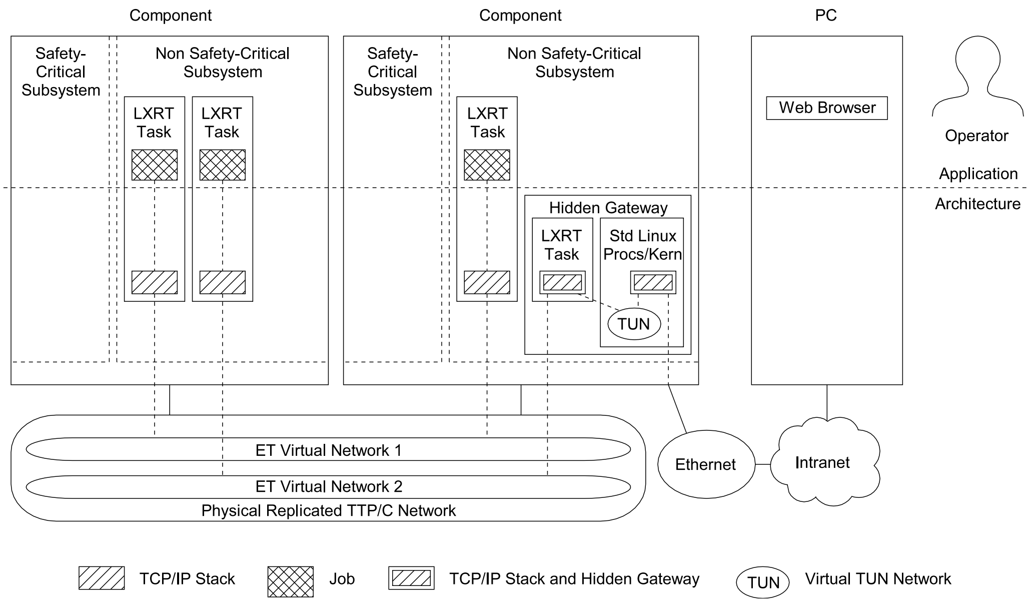


Figure 5.2: Example TCP/IP Network Structure

The Linux kernel Internet Protocol (IP) module is equipped with two local network (in Internet jargon [USC81a]) interfaces, one to the Ethernet and one to a virtual local network (in Internet jargon) inside the Linux kernel (the TUN virtual network). This virtual local network is provided by a Linux kernel module called TUN/TAP device driver [TUN]. It provides a device file to user space and a virtual network interface (the TUN interface) to the kernel IP module. This interface (e.g., called `tun0`) behaves like a standard network interface such as `eth0` (first Ethernet interface). Every IP datagram written to the file in user space is passed to the kernel which receives the datagram at the virtual interface. On the other hand, every IP datagram sent out on the virtual interface by the kernel is written to the file and can be read from the file in user space. As a consequence, the Linux kernel IP module can route datagrams sent from user space, i.e., datagrams sent from the ET virtual network, and destined for the Ethernet and vice versa. On the other hand, the LXRT task's IP module can route those datagrams originating from the virtual TUN interface and destined for the ET virtual network and vice versa. The interface between the LXRT task and the TUN virtual network is implemented by two standard Linux user processes communicating with the LXRT task by shared memory containing queue data structures and semaphores. In Figure 5.2, these two processes can be imagined as the part of the dashed line between the TUN virtual network and the LXRT task which lies inside the right part of the hidden gateway. One process is responsible for the forwarding of IP datagrams from the TUN virtual network to the LXRT task and the other one for the other direction. The former process is signaled by the LXRT task via a semaphore when the LXRT task's receive queue is not full. The latter process is signaled by the LXRT task via a semaphore when the LXRT task wishes to send an IP datagram to the TUN virtual network.

In order to illustrate the assignment of local network interfaces to local networks (in Internet jargon), in Figure 5.3, example IP addresses have been assigned to the local interfaces. In the Figure, the subnetwork mask assigned to all the networks and IP addresses is 255.255.255.0.

Figure 5.4 illustrates an example job of the non safety-critical subsystem containing an HTTP server (Web server) and a UDP application. Below the job, the complex connector unit contains the job's dedicated TCP/IP stack. The job and the TCP/IP stack is implemented in one LXRT task as mentioned above. Below the TCP/IP stack the ETCC module performs the transformation of time-triggered messages to event-triggered messages and vice versa. Below that, the complex connector unit's allocation layer [OPK05] is responsible for statically assigning network resources to the individual jobs. On the physical replicated TTP/C network, two Event-Triggered (ET) virtual networks are established. One is used by the TCP/IP stack of the first job as virtual TCP/IP network and the other one is used for TCP/IP or other event-triggered services by the second job.

In the course of this thesis an IP (Internet Protocol) module, a UDP module, and a minimal ICMP module have been implemented. The author of a related thesis [Ben04] has implemented a TCP stack. In addition, in cooperation with the author of [Ben04], a small HTTP server has been written. These software modules

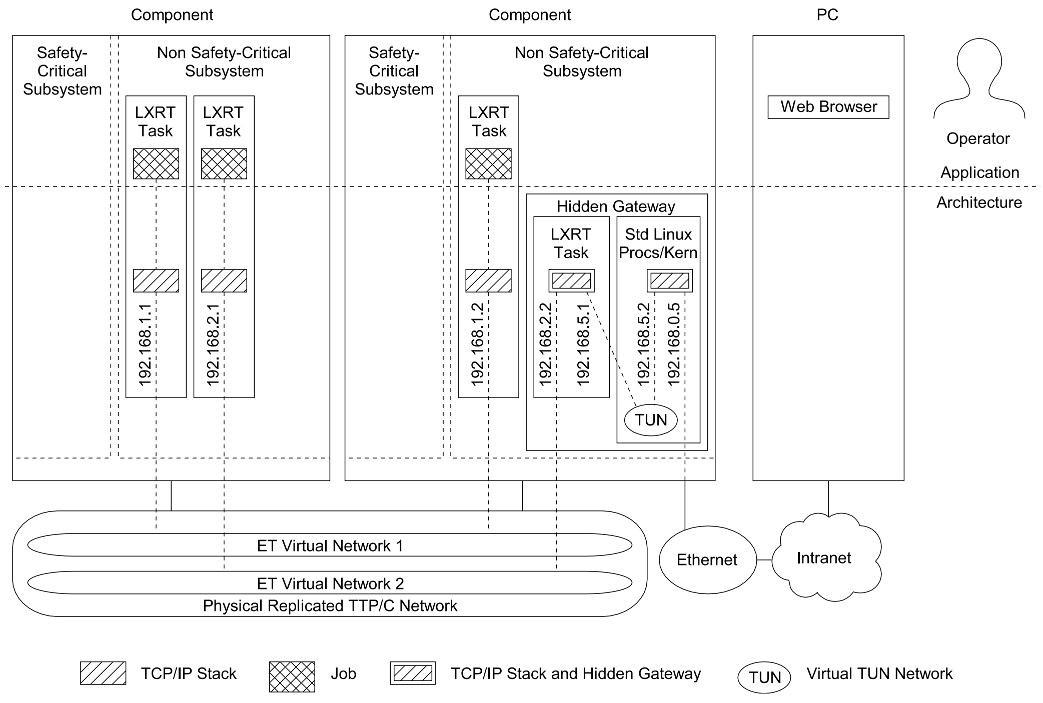


Figure 5.3: Example TCP/IP Network Structure with IP Addresses

(see Figure 5.4) allow to be executed in the protected domain of an LXRT task, i.e., they execute entirely inside periodic execution slots. For example, these modules do not use Linux system calls.

At the link layer, an ETCC module is used for the transport of IP datagrams on ET virtual networks.

At the Internet layer, IP version 4 and ICMP modules providing basic functionality have been implemented. The IP module allows static routing with routing tables editable at run-time. Furthermore, it is possible to define a default gateway [IET89]. IP is implemented without fragmentation, multicasting, and broadcasting. IP datagrams containing IP options are dropped when received and cannot be sent. ICMP is implemented to the extent allowing the answering of ICMP Echo requests (PING service) and sending ICMP "Time Exceeded in Transit" messages.

The transport layer contains the UDP and the TCP modules.

The application layer contains a simple HTTP version 1.1 server allowing the serving of HTML pages. The HTTP server allows the serving of dynamic data (e.g., current time and current memory content) in HTML pages (Figures 5.5 and 5.6). UDP applications can be inserted at this layer as well.

5.3 The Security Gateway

The hidden gateway being the union of the two small hidden gateways in Figure 5.2 contains a security gateway consisting of a stateful packet filtering firewall and a Virtual Private Network (VPN) gateway. This is implemented by using normal Linux iptables [net] for stateful packet filtering and normal Linux IPsec [IPs] for Virtual Private Networking. These two mechanisms allow the protection of the cluster-internal virtual TCP/IP networks from threats originating in external networks such as the Internet.

The VPN gateway is used in order to allow only a limited number of external hosts on the Internet or an intranet to communicate with the cluster. Before communication starts, the gateway or the external host automatically initiates the establishment of an encrypted communication channel (i.e., an IPsec tunnel connecting the external host and the gateway). A common secret key is used in order to authenticate the communicating partners for the key exchange (pre-shared key authentication). The ESP (Encapsulating Security Payload) protocol and a variant of the symmetric Triple DES (Triple Data Encryption Standard) [U.S99] algorithm is used for encrypting the IP datagrams employing the keys exchanged after authentication. In addition to this encryption, the individual IP datagrams are also authenticated by the HMAC-MD5 authentication algorithm. Both encryption and authentication is done in the ESP protocol.

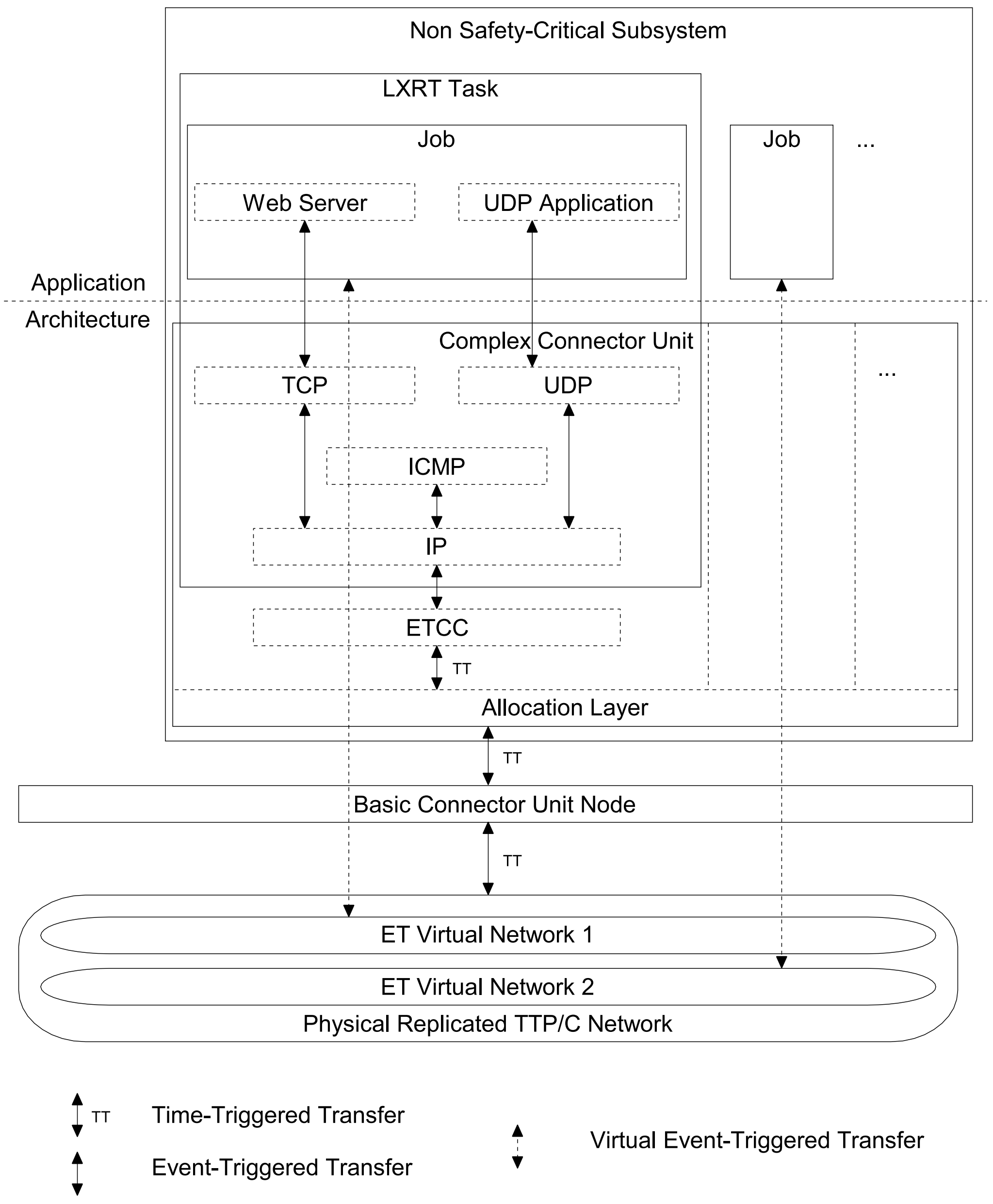
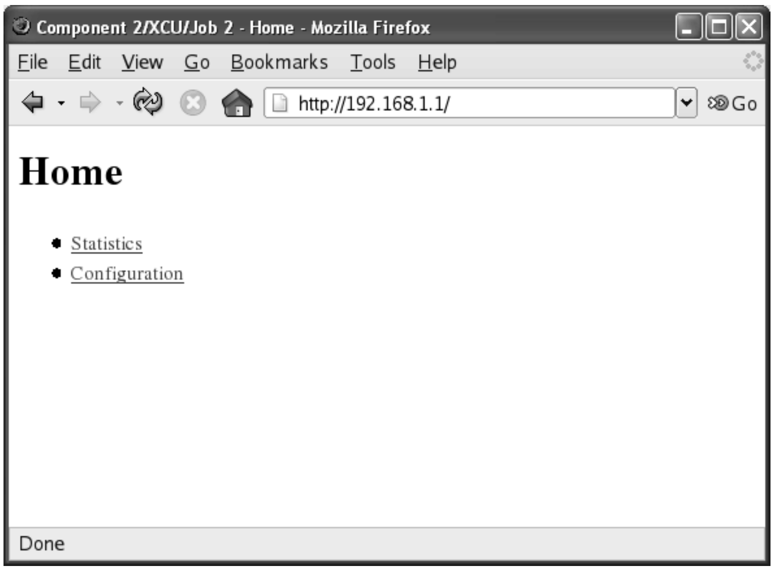


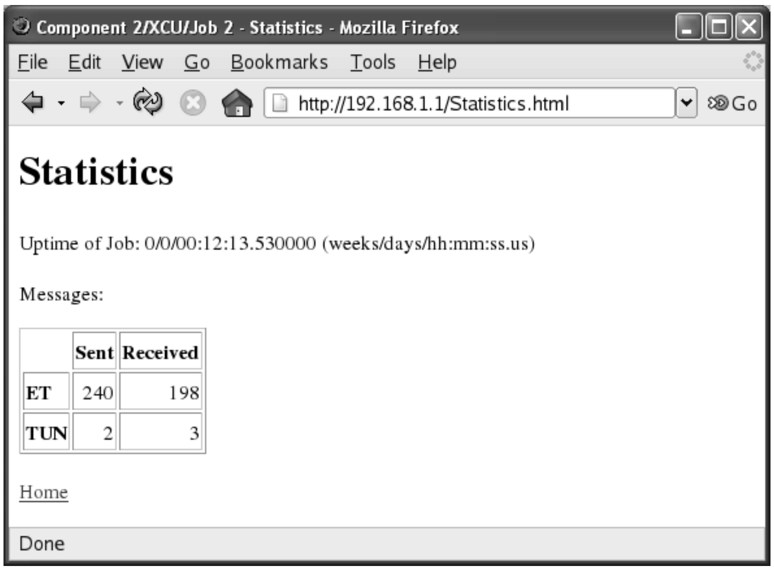
Figure 5.4: Software Modules Dedicated to a Job


Component 2/XCU/Job 2 - Home - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://192.168.1.1/
Go
Done

Home

- Statistics
- Configuration

Figure 5.5: Web server – Home Page


Component 2/XCU/Job 2 - Statistics - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://192.168.1.1/Statistics.html
Go
Done

Statistics

Uptime of Job: 0/0/00:12:13.530000 (weeks/days/hh:mm:ss.us)

Messages:

	Sent	Received
ET	240	198
TUN	2	3

Home

Figure 5.6: Web server – Statistics Page

Chapter 6

Results and Evaluation

In the following, the system model and the implementation is theoretically evaluated. Subsequently, an experimental evaluation of the implementation's security measures is presented.

6.1 Theoretical Evaluation of System Model

The system model employs unified concepts and in part unified fault tolerance mechanisms for non-malicious and malicious faults. Several defensive layers ensure that the probability of intrusions is reduced and especially that an intrusion into the safety-critical subsystem is highly unlikely. The system model mainly aims at tolerating faults by compensation. If faults cannot be tolerated, automatic error detection, automatic error handling, and manual maintenance catches errors. The protection mechanisms allow for the employment of Internet communication services for several security-critical applications.

Unified Concepts for Non-Malicious and Malicious Faults: The concepts of fault-containment region (FCR) and error-containment region (ECR) have been employed in the literature in the context of fault tolerance with respect to non-malicious hardware and non-malicious software faults [Kop03, KOPS04]. In this thesis, these concepts have been applied to several classes of malicious faults such as external physical attacks, external unauthorized service exploitation attacks, external service masquerade attacks, external eavesdropping attacks, external Denial-of-Service (DoS) attacks, subsystem intrusion, and job intrusion. The employment of unified concepts in various areas important in system design might ease the design process.

Unified Fault Tolerance Mechanisms for Non-Malicious and Malicious Faults: System external malicious faults are tolerated by physical protection, authentication (cryptographic authentication and firewall filters), and encryption. System internal malicious faults are tolerated by partitioning and intrusion detection.

Some of the above mechanisms are also applicable in the domain of non-malicious faults. Physical shielding is important for protection from radiation. A firewall and authentication can prevent from accidental system accesses, e.g., by mistyped URLs, IP addresses or TCP/UDP ports. Partitioning is an effective mechanism against non-malicious faults as well [Rus99, KOPS04]. ONAs can be used in order to indicate errors resulting from non-malicious *and* malicious faults [KOPS04].

Several Defensive Layers: [Kop06] applies the Swiss Cheese Model [Rea00] to safety. The Swiss Cheese Model includes several defensive layers (slices of swiss cheese) all of which are imperfect (holes in the slices of cheese). It illustrates that even several defensive layers might allow a hazard to cause a loss if the holes are momentarily aligned to allow the hazard's trajectory to pass through all layers (Figure 6.1).

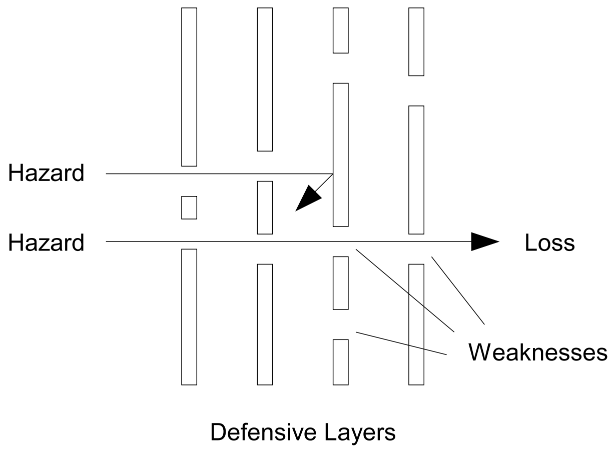


Figure 6.1: Swiss Cheese Model (following [Rea00])

In the system model presented in this thesis, several layers of defense protect the system (Figure 6.2). At first, the firewall protects the cluster-internal network from unauthorized access. Then, the authentication of IP datagrams (e.g., by IPsec) further decreases the probability of intrusion. The authentication at the job's service interface is the next layer of defense. Job partitioning and job intrusion detection provide protection even after a successful attack at the previous layers.

Finally, subsystem partitioning inhibits error propagation from the non safety-critical subsystem to the safety-critical subsystem. The approach of combining security fault prevention with security fault tolerance is proposed in [DR01b].

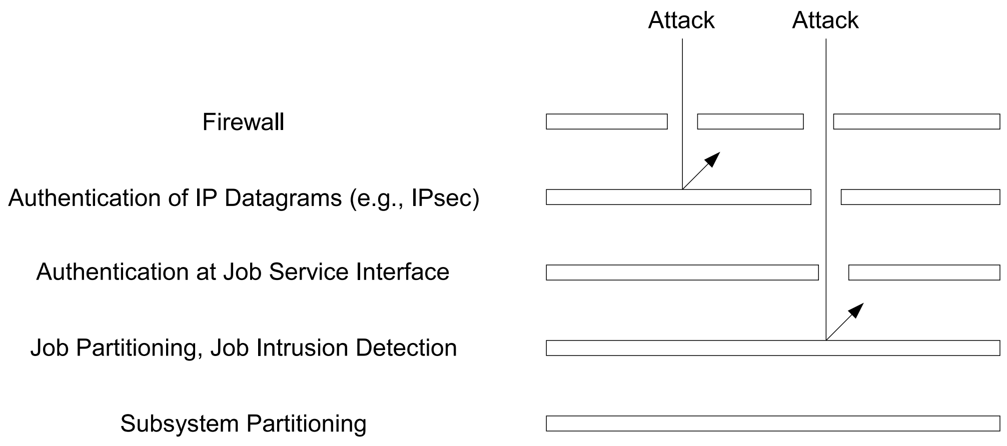

Attack
Attack
Firewall
Authentication of IP Datagrams (e.g., IPsec)
Authentication at Job Service Interface
Job Partitioning, Job Intrusion Detection
Subsystem Partitioning

Figure 6.2: Several Defensive Layers

Fault Tolerance Instead of Maintenance: The use of the concepts of FCR and ECR leads to viewing the considered security problems as fault tolerance problems. These malicious-fault tolerance problems are either problems of attack tolerance or intrusion tolerance.

As a first protective barrier, fault tolerance is provided by compensation with or without explicit error detection (see section 2.1.1), e.g., encryption, encapsulation, etc. In some cases of intrusion, compensation might not be possible, e.g., if a non-replicated job is maliciously modified. Maintenance might be needed in order to return to full system functionality. The difference between fault tolerance and maintenance is that the latter requires the participation of an external agent [ALR01]. For example, after automatic error detection (e.g., signature failure), the maliciously modified job or its DAS (e.g., the navigation system) is automatically deactivated (rollforward to a new correct state). Next, manual fault handling (fault diagnosis, fault isolation, system reconfiguration, and system reinitialization) can be performed (Figure 6.3).

The system model provides fault tolerance as a first layer of protection (see also the section *Several Defensive Layers* above) and offers maintenance rather as a fallback solution.

Protection of the Safety-Critical Subsystem: For the safety-critical subsystem, fault tolerance with respect to malicious faults is provided due to the strong

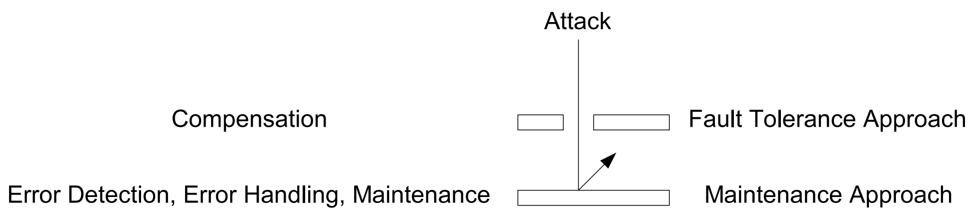


Figure 6.3: Fault Tolerance versus Maintenance Approach

separation of hardware resources and the prohibition of information or control flow from the non safety-critical subsystem into the safety-critical subsystem. The static separation of the safety-critical subsystem and the non safety-critical subsystem at the network-level and at the component-level allows certification of the safety-critical subsystem up to the highest criticality class without paying regard to the non safety-critical subsystem [KOPS04].

Several Applications: The system model paves the way for applications such as condition based maintenance, fault diagnosis, and software updates remotely over the Internet. Engineering feedback from the fleet-in-the-field is possible by continuous communication of the embedded system with the manufacturer's server over the Internet. Moreover, the model allows for multimedia applications and non safety-critical driver assistance exploiting Internet services. A continuous connection to the Internet raises security risks and demands for security mechanisms such as presented in the model. Security is implemented without employing an additional physical network. One physical network is shared securely.

6.2 Theoretical Evaluation of Implementation

In the following, the implementation is compared to the system model and the extent to which the implementation follows the model is outlined.

Direct Physical Access Attacks: Since the implementation uses a DECOS prototype cluster not aiming at the investigation of physical protection mechanisms, the implementation does not consider problems of physical protection.

External Unauthorized Service Exploitation Attacks, External Service Masquerade Attacks, External Eavesdropping Attacks, and External DoS Attacks: In contrast to the system model, the implementation employs an authenticated *and encrypted* VPN IPsec communication channel. The implementation contains no end-to-end encryption (e.g., SSL). As a consequence, the amount of software modules necessary is reduced, i.e., only the security gateway needs to contain authentication and encryption modules. On the other hand, the overall computational load might be increased for scenarios requiring encryption only for a view services instead of all services. However, the implemented authentication averts Denial-of-Service attacks on the cluster-internal virtual TCP/IP networks originating in the Internet. As a first protective barrier, the firewall also impedes such unauthorized access.

Hardware authentication and encryption modules—as proposed in the model in order to prevent from Denial-of-Service attacks on IPsec—have not been added to the implementation. Moreover, the normal Linux Ethernet driver is used. As a consequence, temporal fault propagation from the Ethernet reception module to the node by interrupt-overload should be ruled out by the implementation of a special Ethernet driver (e.g., a “time-triggered” Ethernet driver). Additionally, random number generation in the current implementation might be degraded due to the missing input from the keyboard, mouse, and hard disk on the embedded devices [Ste03, ker].

Node Internal Faults (Intrusions) at Subsystem-Granularity: In the implementation, the separation of the safety-critical subsystem and the non safety-critical subsystem is enforced by the implementation on separate hardware nodes (Figure 5.1). At the network-level, the Ethernet TDMA scheme enforces to some extent the static allocation of network resources to the two subsystems. As a consequence, the safety-critical subsystem is to some extent protected from threats originating from the non safety-critical subsystem or the Internet. For simplicity, the prototype cluster employs Ethernet hubs for intra-component communication. This limits the quality of partitioning. The problem can be solved by employing strongly separated communication channels such as point-to-point links.

Node Internal Faults (Intrusions) at Job-Granularity: Since jobs are executed in Linux user space (in contrast to kernel space), the jobs can exploit Linux memory protection mechanisms. The scheduler [HPOS05] enforces job deadlines and inhibits temporal fault propagation between jobs (e.g., a job’s CPU overload attack on another job). As long as the complex connector unit allocates resources statically to jobs, temporal fault propagation is also averted at the network-level. However, the implementation is vulnerable to attacks exploiting the fact that jobs and the complex connector unit communicate with each other by shared memory not cryptographically protected against access by other jobs. The implementation is in line with the model regarding the assignment of an individual TCP/IP stack to each job. However, the current implementation does not strictly separate the TCP/IP stack from the job's application, i.e., the job's TCP/IP stack and the job execute inside the same LXRT task. The aim of enabling independent development of DASs is reached by the implementation in the sense that compiled job modules can be inserted into the compiled system after compiling the system with empty job modules. Finally, the checking of out-of-norm assertions or intrusion detection services have not been included in the implementation.

6.3 Experimental Evaluation of Implementation

In order to evaluate the security gateway's effectiveness, the Web server has been setup to serve a configuration page, and an FTP server has been installed. Two network vulnerability scanners and a traceroute tool have been employed to identify potential security risks when the security gateway is inactive. In the next step, the scanners and the traceroute tool have been executed while the security gateway—containing the firewall and the VPN gateway—has been active and the results have been compared to the former scanning results. In addition, weaknesses in the configuration containing a firewall only (without IPsec) have been experimentally confirmed.

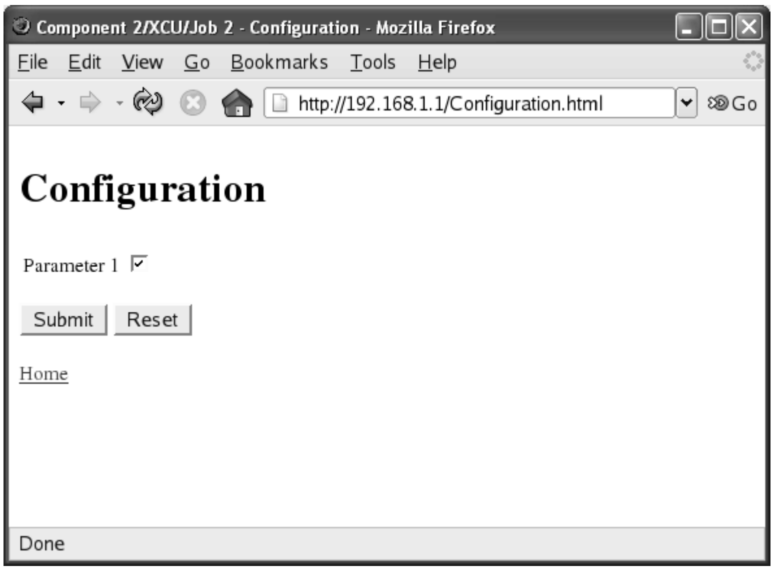
6.3.1 Services

Two exemplary services offered by jobs possessing an Internet interface have been installed in order to enable the evaluation of the security gateway.

Configuration via a Web site

The Web server has been equipped with an HTML form allowing the user to set a configuration parameter inside the job containing the Web server (Figures 6.4 and 6.5). This is accomplished by letting the server accept the HTTP-GET requests "`GET /ConfigurationSet.html?Parameter1=on`" (in order to activate Parameter1) or "`GET /ConfigurationSet.html?`" (in order to deactivate Parameter1).

The cluster-internal virtual TCP/IP network is accessible via static IP (Internet Protocol) routes in the TCP/IP stacks inside the LXRT tasks and inside the Linux kernel as depicted in Figure 6.6.


Component 2/XCU/Job 2 - Configuration - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://192.168.1.1/Configuration.html
Go
Submit
Reset
Home
Done

Configuration

Parameter 1 ☑

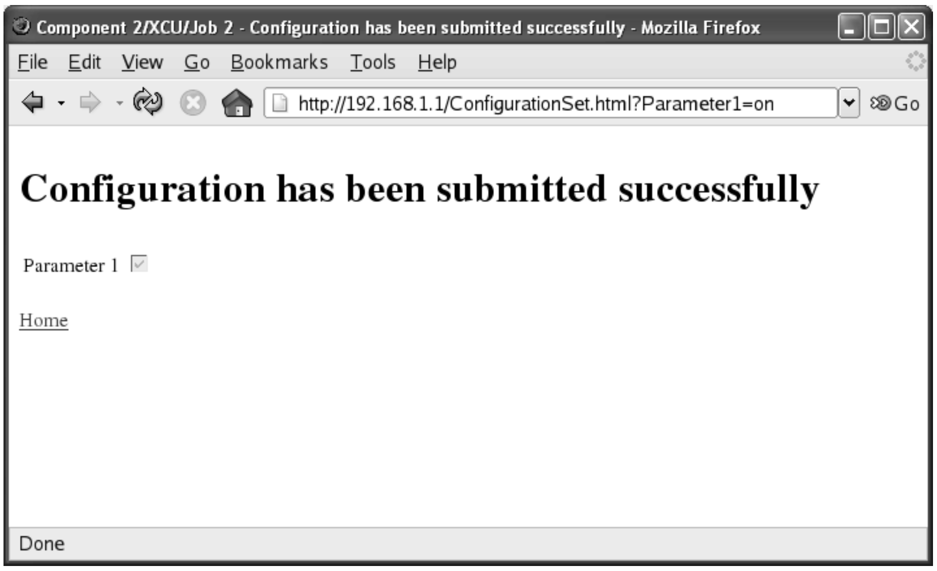
Figure 6.4: Web Page for Configuration

Software Update via FTP

In addition to the Web server, on the same component an open source FTP server has been installed. A precompiled version of the FTP server contained in the Linux distribution Fedora Core 4 (ftpd in package krb5-workstation-1.4-3.i386.rpm) has been used. The FTP server is executed in a normal Linux user process while the Web server is executed inside the LXRT task (Figure 6.7). However, by using an additional TUN virtual network inside the Linux kernel, the LXRT task and the Linux kernel can exchange IP datagrams (on the left in Figure 6.7). As a consequence, the LXRT task's TCP/IP stack has to contain a hidden gateway acting as an IP router. The interface between the LXRT task and the TUN virtual network is the same as described in section 5.2 and also consists of two normal Linux user processes. The implementation of a hidden gateway in the Web server job's LXRT task is only a simplification employed in the course of this evaluation. In order to reach a better approximation of the model, in principle the hidden gateway could also have been implemented inside a separate LXRT task as depicted in Figure 6.8. However, this simplification does not modify any security relevant system properties examined in the following.

6.3.2 Vulnerability Analysis

In order to compare the vulnerabilities of the unprotected system with the protected system, vulnerability scanners have been executed and various kinds of


Component 2/XCU/Job 2 - Configuration has been submitted successfully - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://192.168.1.1/ConfigurationSet.html?Parameter1=on
Go
Done

Configuration has been submitted successfully

Parameter 1 ✓

Home

Figure 6.5: Successful Reconfiguration

attacks have been perpetrated against the system.

The TCP Protocol

At first, a few basic properties of the TCP protocol [USC81b, IET89] (see also section 2.4) relevant to the vulnerability analysis are outlined in the following:

TCP enables two hosts to communicate in both directions by one-way ordered byte (8 bits) streams of arbitrary length. The two streams (one for each direction) are transmitted in many smaller so-called TCP segments usually sent inside IP datagrams. Segment loss, duplication, and reordering are compensated by—among other mechanisms—the inclusion of a sequence number and an acknowledgment number in the TCP segment's header:

In case hosts A and B are communicating by an established TCP connection two one-way data streams can be observed: the stream from A to B ($\mathrm{Stream}_{\mathrm{AB}}$) and the stream from B to A ($\mathrm{Stream}_{\mathrm{BA}}$). Now, some segment s_{AB} sent from A to B contains among other fields a sequence number $\mathrm{SEQ}(\mathrm{s}_{\mathrm{AB}})$ (an integer from 0 to $2^{32}-1$), an acknowledgment number $\mathrm{ACK}(\mathrm{s}_{\mathrm{AB}})$ (also an integer from 0 to $2^{32}-1$), and data bytes $\mathrm{DATA}(\mathrm{s}_{\mathrm{AB}})$ (see Figure 6.9). $\mathrm{SEQ}(\mathrm{s}_{\mathrm{AB}})$ indicates the position of $\mathrm{DATA}(\mathrm{s}_{\mathrm{AB}})$ in $\mathrm{Stream}_{\mathrm{AB}}$. More precisely, sequence numbers count every byte of a stream and $\mathrm{SEQ}(\mathrm{s}_{\mathrm{AB}})$ equals the sequence number of the first byte of $\mathrm{DATA}(\mathrm{s}_{\mathrm{AB}})$. While the sequence number $\mathrm{SEQ}(\mathrm{s}_{\mathrm{AB}})$ refers to $\mathrm{Stream}_{\mathrm{AB}}$, the acknowledgment number $\mathrm{ACK}(\mathrm{s}_{\mathrm{AB}})$ refers to $\mathrm{Stream}_{\mathrm{BA}}$. The acknowledgment number $\mathrm{ACK}(\mathrm{s}_{\mathrm{AB}})$ equals the sequence number of the next byte in $\mathrm{Stream}_{\mathrm{BA}}$ which host A expects to

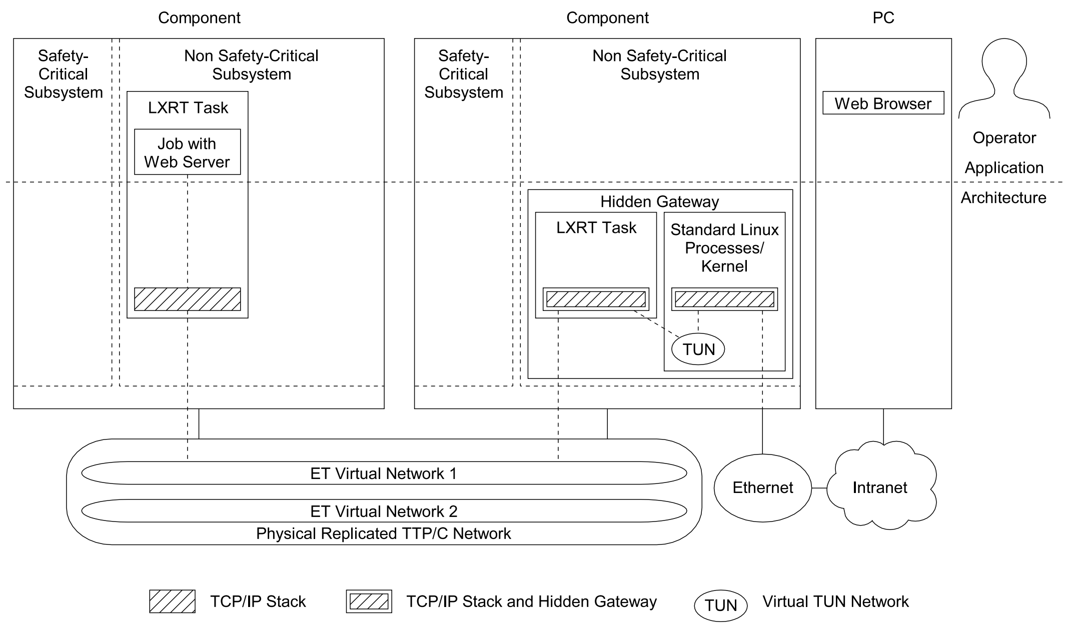


Figure 6.6: Configuration via a Web site

receive. This is agreed to imply that all bytes before that in $\text{Stream}_{\text{BA}}$ have been acknowledged by A. The acknowledgment number of a segment is only meaningful, i.e., it should be processed by the receiver, if the binary ACK (for "acknowledge" meaning "acknowledgment field significant") flag in the segment is set to true.

Normally, a TCP connection between two hosts A and B is established in the following way (see Figure 6.10). At first, one of the hosts, e.g., host B, enters the Listen state in which it waits for connection requests from other hosts.

Then, in the first step of the so-called three-way handshake, host A sends a connection request segment syn_{AB} to host B. In this segment the binary SYN flag (SYN for "synchronize" meaning "synchronize sequence numbers") is set to true. This indicates that $\text{SEQ}(\text{syn}_{\text{AB}})$ equals the so-called initial sequence number (ISN) ISN_{AB} of $\text{Stream}_{\text{AB}}$. This ISN plus 1 modulo 2^{32} is defined to be the sequence number of the first byte in $\text{Stream}_{\text{AB}}$. Since the SYN flag is counted by the sequence numbers as if it were one data byte, ISN_{AB} is not assigned to the first data byte of $\text{Stream}_{\text{AB}}$ but rather only to the SYN flag. Only the next sequence number $((\text{ISN}_{\text{AB}} + 1) \bmod 2^{32})$ is assigned to the first data byte in $\text{Stream}_{\text{AB}}$.

In the second step of the three-way handshake, host B replies by sending a segment $\text{synack}_{\text{BA}}$ having the ACK and the SYN flags set. $\text{ACK}(\text{synack}_{\text{BA}})$ equals $(\text{ISN}_{\text{AB}} + 1) \bmod 2^{32}$, i.e., ISN_{AB} received in the first step is acknowledged by B. Moreover, $\text{SEQ}(\text{synack}_{\text{BA}})$ equals ISN_{BA}.

In the final third step of the three-way handshake, A acknowledges ISN_{BA}

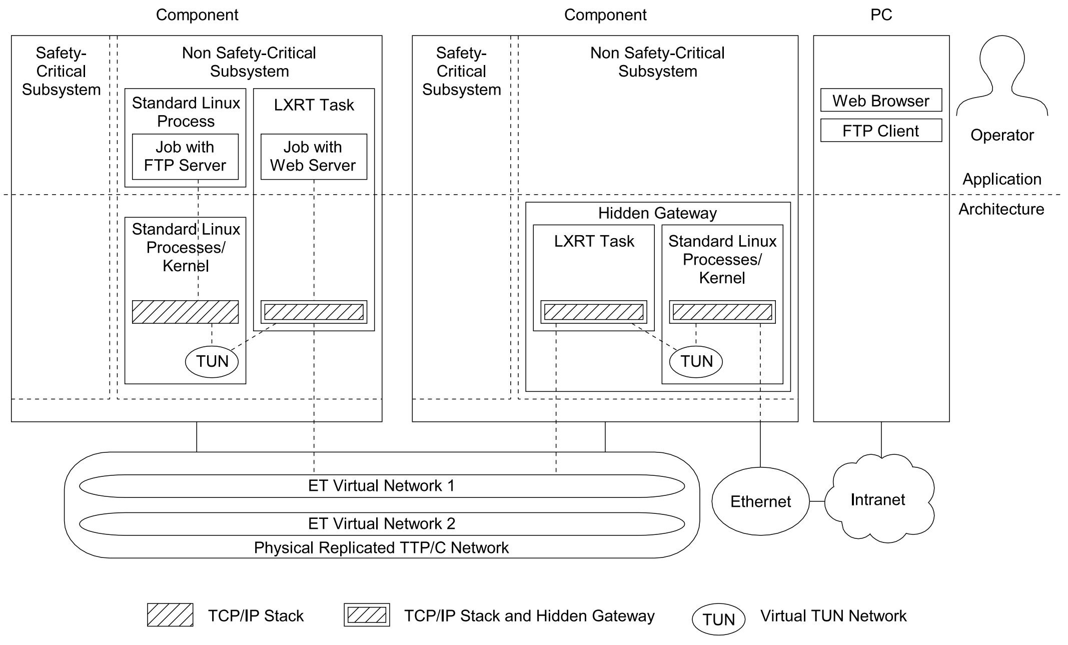


Figure 6.7: Software Update via FTP

received in the second step by sending a segment ack_{AB} with the ACK flag set and $ACK(ack_{AB})$ equal to $(ISN_{BA} + 1) \bmod 2^{32}$.

In addition to the ACK and the SYN flag, the FIN flag (FIN for “finis” meaning “no more data from sender”) and the RST flag (RST for “reset” meaning “reset the connection”) are of relevance to the vulnerability analysis. A segment containing the FIN flag indicates to the receiver of the segment that the sender has no more data to send, i.e., the stream from sender to receiver is ended. The standard mechanism of closing a connection completely (both streams) is that both hosts send a FIN segment. After the FIN segments have been acknowledged, the connection data structures can be deleted on both sides. On the other hand, upon reception of a correct segment with the RST flag set, the connection data structures are deleted without further communication with the other host, i.e., the connection is aborted. For example, this is used if one of the hosts crashes and after reboot receives further segments from the other side. In response, the host will send a RST segment indicating that the connection does not exist anymore. As a consequence, the receiver of the RST segment will abort the connection and delete its connection data structures. Another possibility of needing to send a RST segment is that a connection abort is initiated by the TCP user, e.g., some application.

Another important TCP mechanism is the transmission of the so-called receive window in TCP segments for implementing flow control. The receive window is

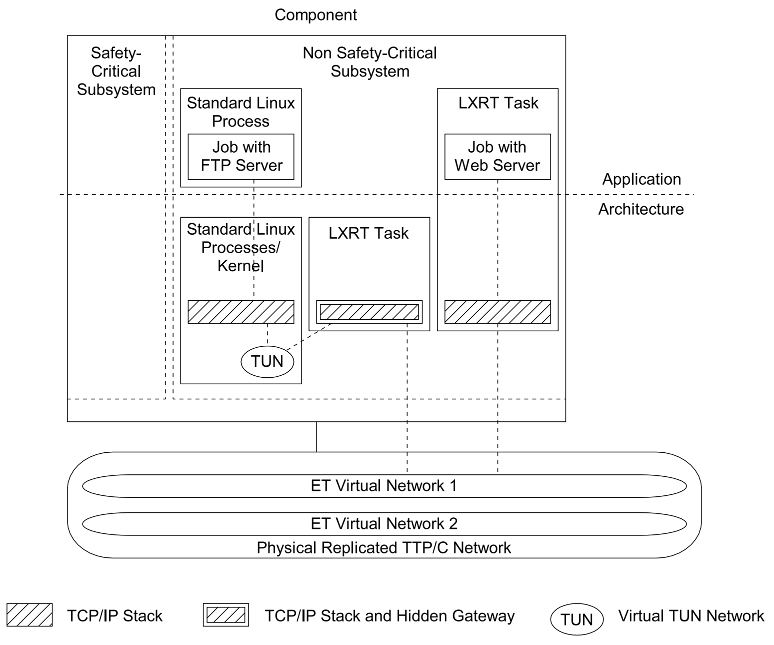


Figure 6.8: Separation of Hidden Gateway and TCP/IP Stack of Job

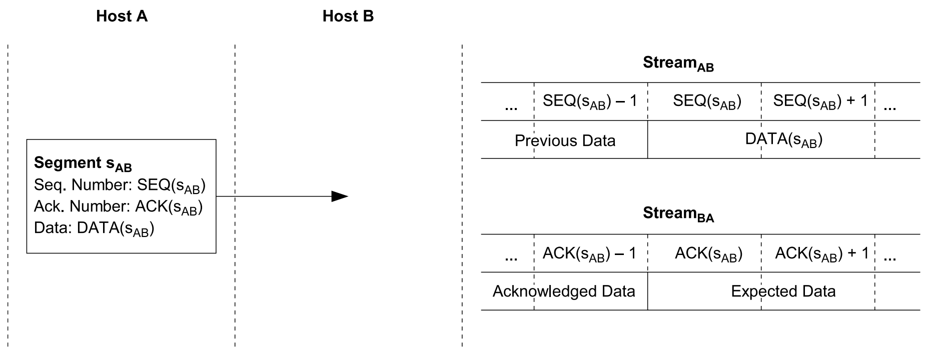


Figure 6.9: TCP Sequence Number and Acknowledgment Number Fields

an unsigned integer with different ranges depending on the standard. The TCP standard [USC81b, IET89] of the highest maturity level ("Standard") defines the receive window to be an integer in the range from 0 to $2^{16}-1$. On the other hand, a TCP standard update [JBB92] currently at the first ("Proposed Standard") of three maturity levels introduces a "Window Scale Option" to be inserted in the TCP segment header. This option expands the receive window range to the larger integer set $\{m2^n | m \in \{0,1,\ldots,2^{16}-1\}, n \in \{0,1,\ldots,14\}\}$. Two Linux versions employed in the evaluation (the Linux version executing the scanners and the Linux version executing the FTP server on the Soekris node) make use of this option. The receive window contained in a segment tells the segment's receiver how many data bytes the segment's sender is willing to receive counted from the

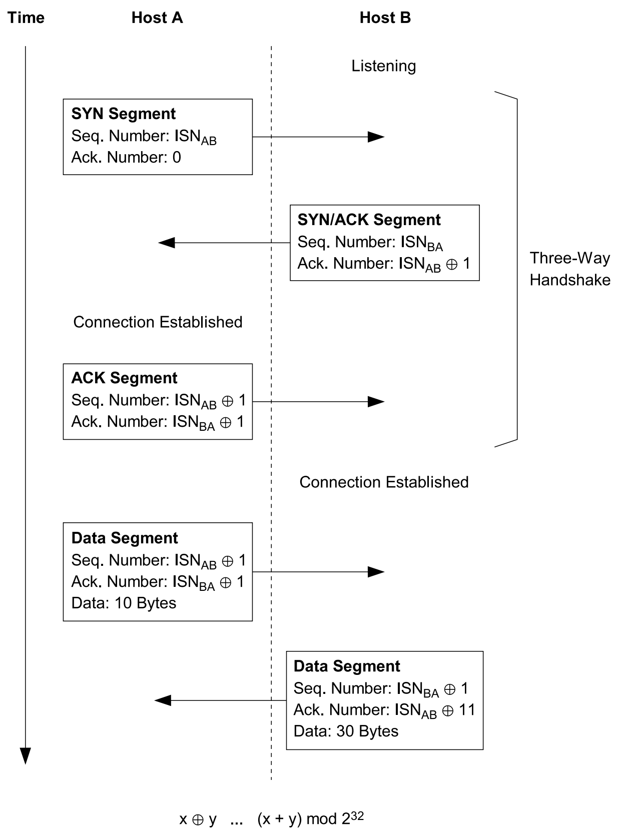

Time
Host A
Host B
Listening
SYN Segment
Seq. Number: ISN_{AB}
Ack. Number: 0
SYN/ACK Segment
Seq. Number: ISN_{BA}
Ack. Number: $ISN_{AB} \oplus 1$
Three-Way Handshake
Connection Established
ACK Segment
Seq. Number: $ISN_{AB} \oplus 1$
Ack. Number: $ISN_{BA} \oplus 1$
Connection Established
Data Segment
Seq. Number: $ISN_{AB} \oplus 1$
Ack. Number: $ISN_{BA} \oplus 1$
Data: 10 Bytes
Data Segment
Seq. Number: $ISN_{BA} \oplus 1$
Ack. Number: $ISN_{AB} \oplus 11$
Data: 30 Bytes
$x \oplus y$ ... $(x + y) \bmod 2^{32}$

Figure 6.10: TCP Three-Way Handshake

byte assigned to the sequence number contained in the segment's acknowledgment number field. This allows a TCP receiver to reduce the amount of data sent to it if, e.g., the receive buffers are close to full.

Vulnerability Scans

The vulnerability scanners have been Nessus [Nes] version 2.2.5, SARA (Security Auditor's Research Assistant) [SAR] version 6.0.7a, and the traceroute tool from the Fedora Core 4 package traceroute-1.4a12-26.i386.rpm. These tools have been executed on a PC residing on the same private local Ethernet network as the cluster (Figure 6.11).

The traceroute results (i.e., the results showing the IP routes to a destination IP address) have been taken from the traceroute tool (with the -I option, i.e., ICMP Echo messages are sent) and not from the other scan results since the traceroute tool's results are more expressive. Furthermore, only for the traceroute experiments (i.e., the execution of the traceroute tool) the TCP/IP stacks of the LXRT tasks have been equipped with the function of returning ICMP "Time

Exceeded in Transit" messages. This has been in order to demonstrate traceroute possibilities in standard configurations. However, this functionality is not needed for successful communication via TCP/IP. "Time Exceeded in Transit" messages are sent by the ICMP module if a received IP datagram should be forwarded but has a Time-to-Live (TTL) value of 0 or 1. This fact is exploited by the traceroute tool by sending datagrams with increasing TTL values. This triggers "Time Exceeded in Transit" messages from intermediate routers.

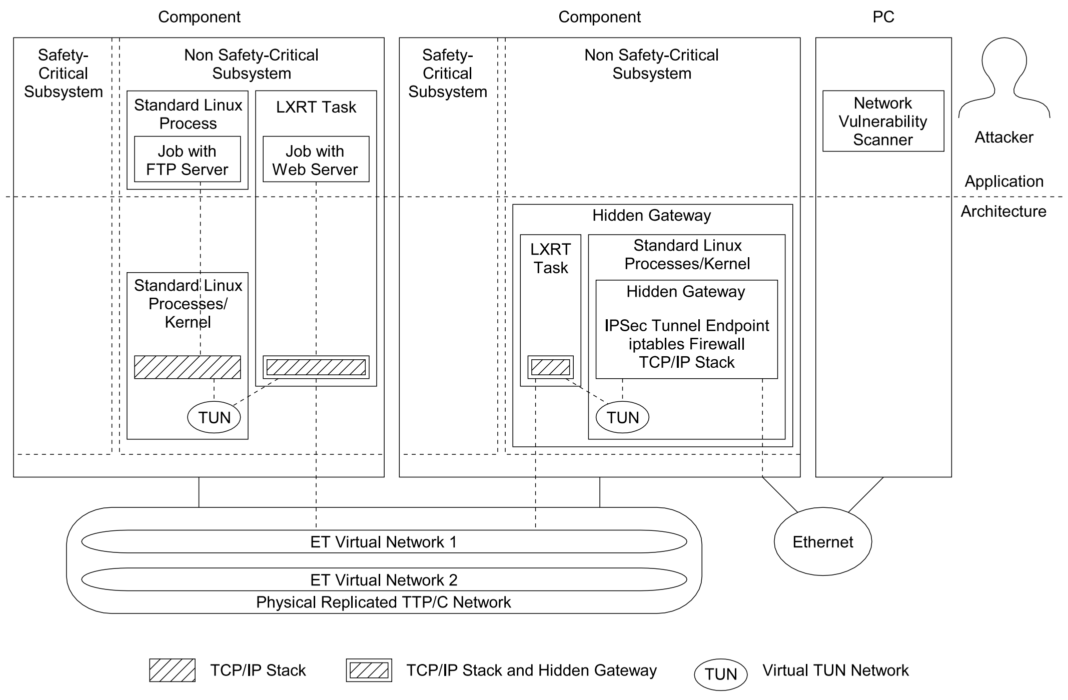


Figure 6.11: The Security Gateway

Some scan results returned by the scanners have been incorrect. This can occur, e.g., if packets are lost due to network overload when many tests are carried out simultaneously. Only results confirmed by the author of this thesis are included below. The scan results are not reproduced verbatim but try to reflect the meaning of the results as exactly as possible.

Vulnerability Scans with Security Gateway Deactivated: In the first series of experiments the security gateway is deactivated. In its place resides only a router (inside the Linux TCP/IP stack of the hidden gateway to the right in Figure 6.11) forwarding all packets. The cluster-internal virtual TCP/IP network is fully accessible from the PC executing the vulnerability scanners. Figure 6.12 is a copy of Figure 6.11 with the addition of the IP addresses assigned to the local network (in Internet jargon [USC81a]) interfaces. All IP addresses and local IP networks have the subnetwork mask 255.255.255.0. Each scan is addressed against one IP address. These are the results of the scans:

1. FTP server job (192.168.55.2):

   (a) An FTP server is running on TCP port 21. It is possible to obtain the banner of the FTP server by connecting to it. This is the FTP server's banner:
   220 (none) FTP server (Version 5.60) ready.

   (b) The host is running Linux kernel 2.6.

   (c) It may be possible to bypass firewall rules. The host does not discard TCP SYN segments which have the FIN flag set. Depending on the kind of firewall used, an attacker may use this flaw to bypass its rules.

   (d) The host might be vulnerable to a sequence number approximation bug, which may allow an attacker to send spoofed RST packets to the host and close established connections. This may cause problems for some dedicated services (BGP, a VPN over TCP, etc.).

   (e) It is possible to determine the exact time set on the host. The host answers to an ICMP timestamp request. This allows an attacker to know the time which is set on your machine. This may help him to defeat all your time based authentication protocols.

   (f) The traceroute to the host is:
   192.168.0.70 → 192.168.0.57 → 192.168.57.1 → 192.168.1.1 → 192.168.55.2

   (g) It is possible to force the FTP server to connect to third-party hosts by using the PORT command. This problem allows intruders to use your network resources to scan other hosts, making them think the attack comes from your network, or it can even allow them to go through your firewall.

   (h) The FTP server allows users to make any amount of PASV commands, thus blocking the free ports for legitimate services and consuming file descriptors.

   The last two results have only been returned when the respective scanner has been supplied with a valid FTP username and password. In practice this might be relevant if an attacker found out an FTP username and password by inspecting packets on the local network. (FTP usernames and passwords are transmitted in clear.)

2. Web server job (192.168.1.1):

   (a) Port 80 (HTTP) offers a service.

   (b) The following CGIs have been discovered:
   Syntax:
   `cginame (arguments [default value])`
   `/ConfigurationSet.html (Parameter1 [])`
   (CGI (Common Gateway Interface) [CGI] is a standard protocol for information servers such as Web servers allowing the remote execution of server side programs.)

   (c) The traceroute to the host is:
   192.168.0.70 $\rightarrow$ 192.168.0.57 $\rightarrow$ 192.168.57.1 $\rightarrow$ 192.168.1.1

3. Security gateway (192.168.0.57):

   (a) It is possible to determine the exact time set on the host. The host answers to an ICMP timestamp request. This allows an attacker to know the time which is set on your machine. This may help him to defeat all your time based authentication protocols.

   (b) The traceroute to the host is:
   192.168.0.70 $\rightarrow$ 192.168.0.57

Discussion: The scanners have detected the FTP server and the Web server since the servers accept connections from arbitrary locations. The visibility of the FTP server's banner helps attackers to exploit well known vulnerabilities of popular FTP servers. Similarly, the detectability of the operating system version used (here Linux kernel 2.6) supports attackers.

The TCP vulnerability allowing also SYN segments which have the FIN flag set, is problematic in case a firewall has a vulnerable method of checking TCP flags. TCP binary flags are encoded as individual bits inside the header bytes. A firewall might store all flags inside one integer variable, e.g., the flag sequence `00010011` might be stored as 8-bit unsigned integer 19 ($= 2^4 + 2^1 + 2^0$). If the firewall compares only this variable to a variable representing a forbidden flag sequence instead of comparing individual bits, it could allow a TCP segment with the SYN and the FIN flag set while it would drop the same packet without the FIN flag set. In most cases, the firewall user intends to drop both kinds of TCP

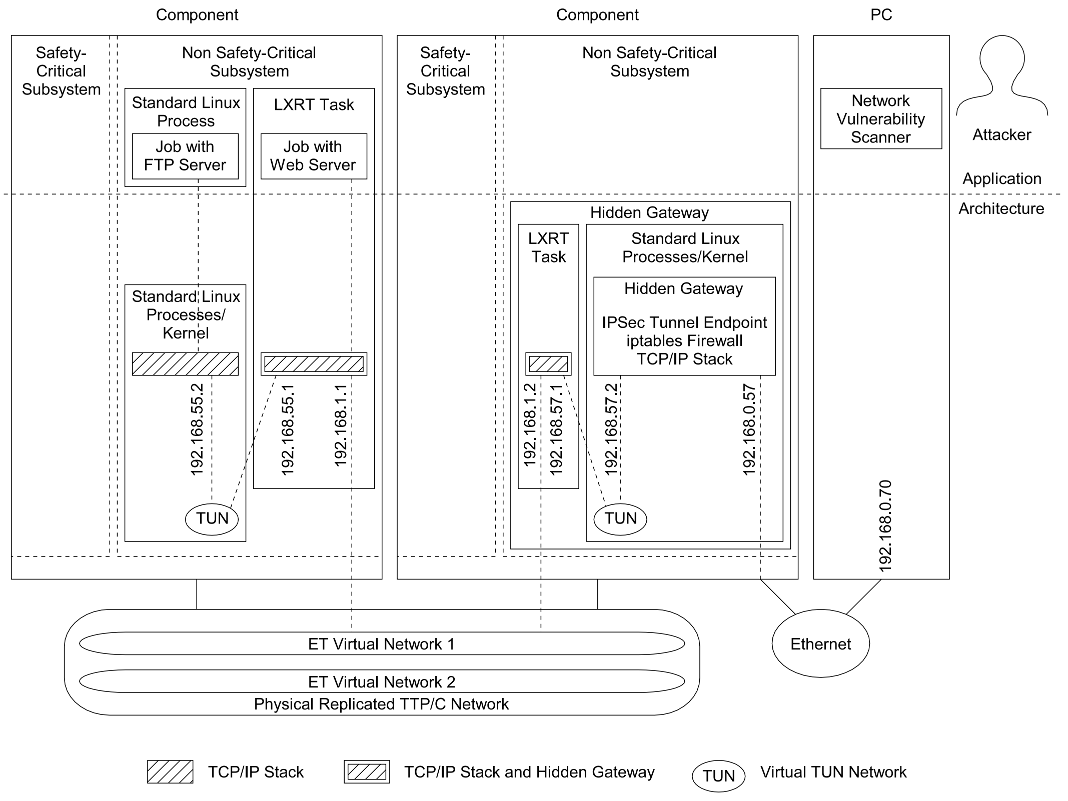


Figure 6.12: IP Addresses Assigned in Evaluation Scenario

segments. Such a kind of firewall—in combination with this TCP vulnerability—might allow an attacker to bypass the firewall by sending TCP SYN segments with the FIN flag set [USC].

The sequence number approximation bug references the problem of forged TCP RST segments injected into an established TCP connection with the malicious intent to close it prematurely [Wat04a, NVD]. In the case of an established connection, the TCP standard [USC81b] defines a RST segment to be acceptable if its sequence number is in the receiver's receive window. Although the standard mentions the checking of the acknowledgment number field as an additional option in deciding whether or not to accept a RST segment, [Wat04b] (the presentation slides to [Wat04a]) says that no TCP module tested by the author of the slides verified the acknowledgment number field. Moreover, in an experiment, conducted by the author of this thesis, also the TCP module of the Linux version executing the FTP server did not seem to verify the acknowledgment number field in a RST segment. As a consequence, probably in several cases, a forged RST segment can be injected into an established connection by sending a TCP segment with the connection's source and destination ports and an *arbitrary* sequence number in the receiver's receive window. The TCP segment has to be encapsulated inside an IP datagram with the connection's source and destination IP addresses. This injection of a RST segment into an established connection is possible from any place on the Internet if networks between the attacker and the target do not employ appropriate protection mechanisms, e.g., ingress and egress filters (see subsection *External Denial-of-Service Attacks* of section 4.3.2). As outlined in [Wat04a], the attack might be simplified substantially if the "Window Scale Option" is used at the target and the target employs large windows. The larger the windows, the smaller the range of sequence numbers that has to be traversed. Approaches to guessing both TCP ports of the connection are outlined in [Wat04a, Wat04b].

In the course of this evaluation, an experiment has been conducted in order to confirm this vulnerability for the TCP module of the Linux version hosting the FTP server: at first, a connection has been established to the FTP server. Then, a TCP RST segment with the sequence number "receive window start" plus 5700 (the receive window has been 5792) has been generated by a tool and sent to the FTP server. Finally, a legal segment sent to the server has been rejected by the TCP module with a RST segment indicating the success of the attack.

The vulnerability scanner's result mentions BGP (Border Gateway Protocol) and VPNs over TCP as services vulnerable to these RST attacks. This is probably only due to the fact that these services employ longer lasting TCP connections (this allows an attacker to try sequence numbers for a longer period of time) and at least BGP employs larger windows than some other services [NVD].

The ICMP timestamp message might pose a risk to the system if pseudo-random number generators are seeded with the current time [CVEc] and are used for securing the system, e.g., for encryption. [GW96] explains how in 1996 a version of Netscape's Web browser has been found vulnerable due to poor random number generation (employing the current time and other factors) in the SSL module.

The correct routes to the destination IP addresses have been discovered in all three cases (see Figure 6.12). This shows that the structure of the cluster-internal virtual TCP/IP networks might be revealed to external hosts if no precaution is taken.

The FTP PORT command is sent to FTP servers by FTP clients in order to inform the server about the IP address and the TCP port the server should connect to for the FTP data connection. This allows an intruder to initiate connections from the FTP server to arbitrary other locations. For example, a firewall can be circumvented for scanning the ports of host A if the FTP server and host A reside on the same subnet behind the firewall: the attacker uses the PORT command to instruct the FTP server to connect to a port on host A. In case the FTP server indicates that the data connection has been established successfully the attacker concludes that the port is open. On the other hand, if the FTP server indicates an error, the port might be closed or protected by additional means. An additional advantage of this port scanning technique is that the port scan seems to originate from the FTP server's host and not from the attacker's host [CVEa, CER].

The FTP PASV (passive) command is issued by a client in order to instruct the FTP server to start listening for a data connection. After the reception of the PASV command the FTP server chooses a port to listen on, starts listening on the port for incoming connections, and sends the port number to the client. Now, the client initiates the data connection from the client to the port chosen by the server. If the FTP server allows clients to issue too many PASV commands, an attacker who has access to at least one FTP account is able to launch a DoS attack on other FTP users by executing sufficiently many PASV commands [CVEb]. In the case of this evaluation, it was possible to issue approximately one thousand PASV commands per user session. It has been sufficient to log in five times simultaneously as the same user and to issue one thousand PASV commands per session. After this, another user has been able to log in to the FTP server but has not been able to successfully launch the PASV command any more since the FTP server has tried to listen on a port which has been occupied already. Since this experiment involves a high communication load, it has been carried out employing a direct Ethernet connection to the node containing the FTP server in contrast to the connection through the ET virtual network. This has been done only due to the limited bandwidth of the ET virtual network in the prototype configuration and should not weaken the result of the experiment.

The Web server has been identified to accept a CGI-like request allowing the passing of a parameter. The scanner has been able to identify the correct URL (`/ConfigurationSet.html?Parameter1=...`). On the other hand, it has failed to identify possible values to pass to the variable `Parameter1`. However, this allows an attacker to at least try some values for the variable `Parameter1` or to infer the correct value from the HTML source of the configuration Web page in order to possibly cause damage to the job's state. In the case of this evaluated system, an attacker might easily succeed in causing harm to the job's state because the Web server checks only the existence of the string `Parameter1=on` in the URL. If the string exists, the parameter is set to true. If the string is not contained in the URL, the parameter is set to false.

The confirmation experiments regarding the SYN/FIN and the RST vulnerabilities have been conducted after the addition of the ICMP “Time Exceeded in Transit” functionality to the TCP/IP stacks in the LXRT tasks. This is assumed not to have altered the outcome of the confirmation experiments.

Vulnerability Scans with Security Gateway Activated: For the following experiments, the security gateway’s firewall has been instructed to block all packets. Exceptions as described in the following have been specified. The firewall has been configured to allow ICMP ping requests (from anywhere to Web server and FTP server jobs, from anywhere to security gateway) and replies (from Web server and FTP server jobs to anywhere) in order to let the scanners “see” the scanned targets. Moreover, TCP connections to the FTP server including related connections (e.g., FTP data connections) and TCP connections to the Web server from the legitimate host (here the IP address 192.168.0.71 has been used) have been allowed by entering appropriate rules in the firewall’s configuration file. In addition, at the security gateway, UDP destination port 500 which is used by IPsec has been kept open in the firewall configuration and the IPsec protocol ESP has been allowed to be received by the firewall. The sending of packets from the security gateway has not been restricted.

The IPsec module at the security gateway has been instructed to tunnel all traffic to and from the legitimate host to the jobs containing the FTP server and the Web server through an encrypted and authenticated IPsec tunnel. Before the scans have been started, the IPsec tunnel has been tested by assigning the IP address 192.168.0.71 (legitimate host) to the PC in Figure 6.12. Then the PC has been reconfigured to another IP address (192.168.0.70) simulating a malicious network participant. From this IP address the scans have been launched against the embedded network. In the following, the results of these scans are listed:

1. FTP server job (192.168.55.2):
   (a) The traceroute to the host is:
   192.168.0.70 $\rightarrow$ 192.168.0.57 $\rightarrow$? $\rightarrow$? $\rightarrow$ 192.168.55.2
2. Web server job (192.168.1.1):
   (a) The traceroute to the host is:
   192.168.0.70 $\rightarrow$ 192.168.0.57 $\rightarrow$? $\rightarrow$ 192.168.1.1
3. Security gateway (192.168.0.57):
   (a) The traceroute to the host is:
   192.168.0.70 $\rightarrow$ 192.168.0.57

Discussion: Now, the routes to the destination IP addresses have not been discovered since the firewall has blocked ICMP "Time Exceeded in Transit" messages sent by intermediate routers. The addresses listed in the results are only its own (192.168.0.70), the security gateway's (192.168.0.57), and the destination's IP addresses. The other addresses have not been discovered (indicated by question marks). In the evaluation scenario, the security gateway's IP address is known to the scanner anyway since the security gateway is the first intermediate router connecting the scanner to the cluster.

The absence of all other scan results shown in the first list of results (scans with the security gateway deactivated) indicates that the security gateway has successfully protected the cluster from the scanner. Furthermore, no new scan results not found before have been returned by the scanners. As a consequence, the activation of the firewall and the IPsec module does not seem to have added new vulnerabilities.

In the following, experiments demonstrating weaknesses in the configuration containing an active firewall only (without IPsec) are presented in order to confirm the necessity of additional protection mechanisms such as authentication, e.g., IPsec.

External Unauthorized Service Exploitation Attacks

The first attack presented, in principle can be conducted from anywhere on the Internet, i.e., remotely or off-path [OB06] (Figure 6.13). (The path referred to in the term off-path is the communication path between the cluster and the legitimate host, i.e., the host allowed by the firewall rules to communicate with the cluster.) The second attack can only be executed on the local network (in Internet jargon) to which the legitimate host is connected (Figure 6.14).

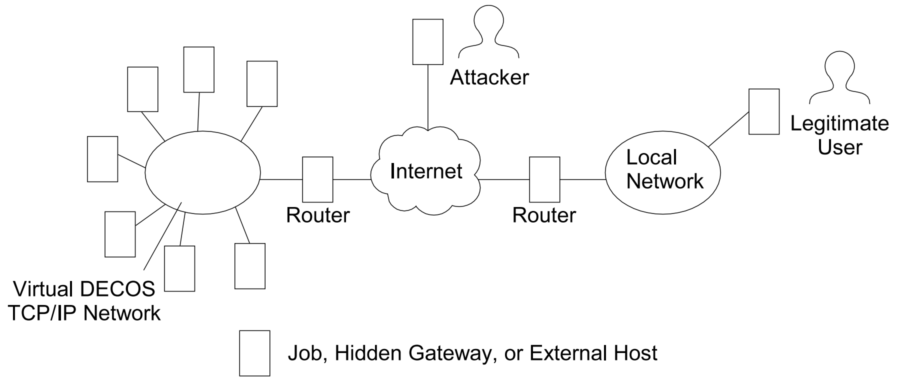

Attacker
Legitimate User
Local Network
Internet
Router
Router
Virtual DECOS TCP/IP Network
Job, Hidden Gateway, or External Host

Figure 6.13: Remote Attack

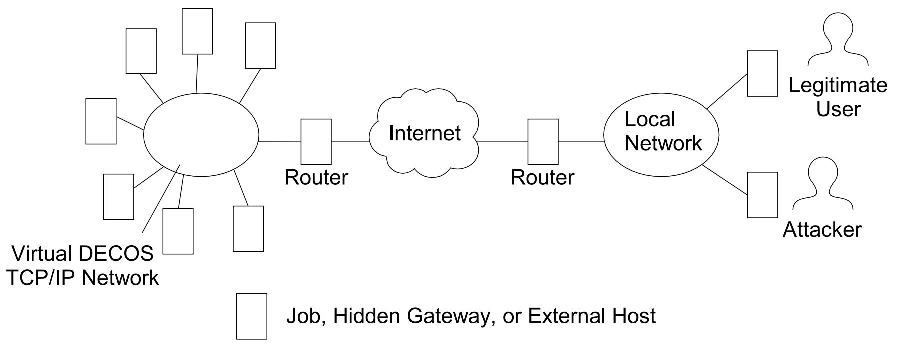

Internet
Local Network
Router
Router
Legitimate User
Attacker
Virtual DECOS TCP/IP Network
Job, Hidden Gateway, or External Host

Figure 6.14: Local Attack

An important mechanism enabling the circumvention of a firewall is TCP/IP spoofing [HM96]. IP spoofing is concerned with forging an IP datagram's header in order to pretend that another IP source host originated the datagram. In the IP datagram header's IP source address field the IP address of a legitimate host allowed by the firewall rules is inserted instead of the correct IP source address (i.e., the attacker's IP source address). In case the networks between the attacker and the target host (receiver of IP datagram) do not implement any protection mechanisms, e.g., ingress and egress filters (see subsection *External Denial-of-Service Attacks* of section 4.3.2), the IP datagram should reach its destination even though the IP source address is incorrect. The receiver and its firewall cannot distinguish the datagram from an authentic datagram. In Figure 6.15, at first an authentic datagram is sent to host B and then a forged datagram is sent from the attacking host C.

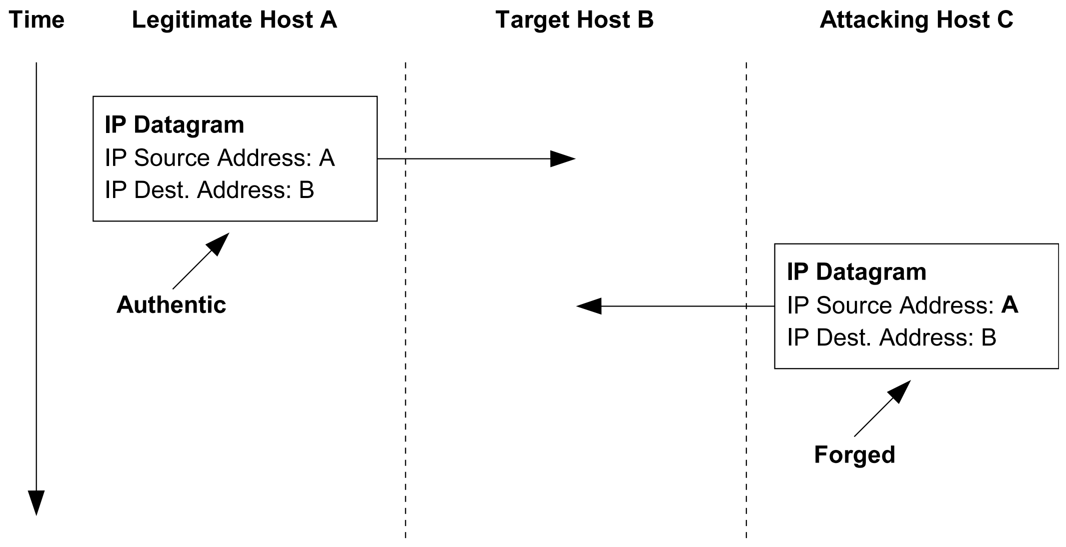


Figure 6.15: IP Spoofing

Remote Attack: In addition to forging data in the IP header (IP spoofing), also data in a TCP segment's header might be forged in order to attack services employing the TCP protocol. Figure 6.16 illustrates a scenario in which the attacking host convinces the target host that the TCP segments sent from the attacking host to the target host originate at another legitimate host. At first, a SYN segment encapsulated in an IP datagram with a forged IP source address is sent. The TCP header does not have to contain forged data. Next, the target host responds by a SYN/ACK segment destined for legitimate host A. Host A is assumed to be not responding, e.g., because it is not online or currently under a packet flooding attack by host C. If host A were responding, it could send a RST segment aborting the connection. Since the attacking host does not receive the initial sequence number (ISN) ISN_{BA}, it has to predict it for inclusion in the next segment. This can be done if the target generates predictable ISNs, e.g., ISNs increasing by a fixed number every millisecond. Now, the ACK segment containing the acknowledgment number $(\text{ISN}_{\text{BA}} + 1) \bmod 2^{32}$ (in the figure denoted as $\text{ISN}_{\text{BA}} \oplus 1$) derived from the predicted sequence number ISN_{BA} is sent to the target. Next, if the sequence number has been predicted correctly, the target believes that a connection has been established between host A and itself. Finally, the attacking host sends a TCP segment containing data to the target which accepts it as authentic and originating from host A.

In the course of this evaluation, a TCP/IP spoofing attack against the Web server job has been perpetrated while the firewall (without IPsec) of the security gateway has been active (Figure 6.12). The attacking host has been the PC at IP address 192.168.0.70. In order to allow the attacking host to predict ISNs, a dummy SSH server listening at TCP port 22 for connection requests has been installed inside the Web server job. The dummy SSH server has been instructed to close every connection to it right after establishment. The firewall has been configured to allow any host to connect to the dummy SSH server. In practice, this scenario could occur because it might be assumed that the SSH server is secure anyway and does not have to be protected by the firewall.

The experiment starts by sending a SYN segment to the SSH server and remembering the ISN returned. The connection is aborted by sending a RST segment. One second after the first SYN segment the experiment proceeds as outlined in Figure 6.16. First, the connection request, i.e., a SYN segment containing a legal IP source address (allowed by the firewall), is sent to the target. After the target presumably has sent the SYN/ACK segment, an ACK segment containing the predicted ISN plus 1 modulo 2^{32} in the acknowledgment num-

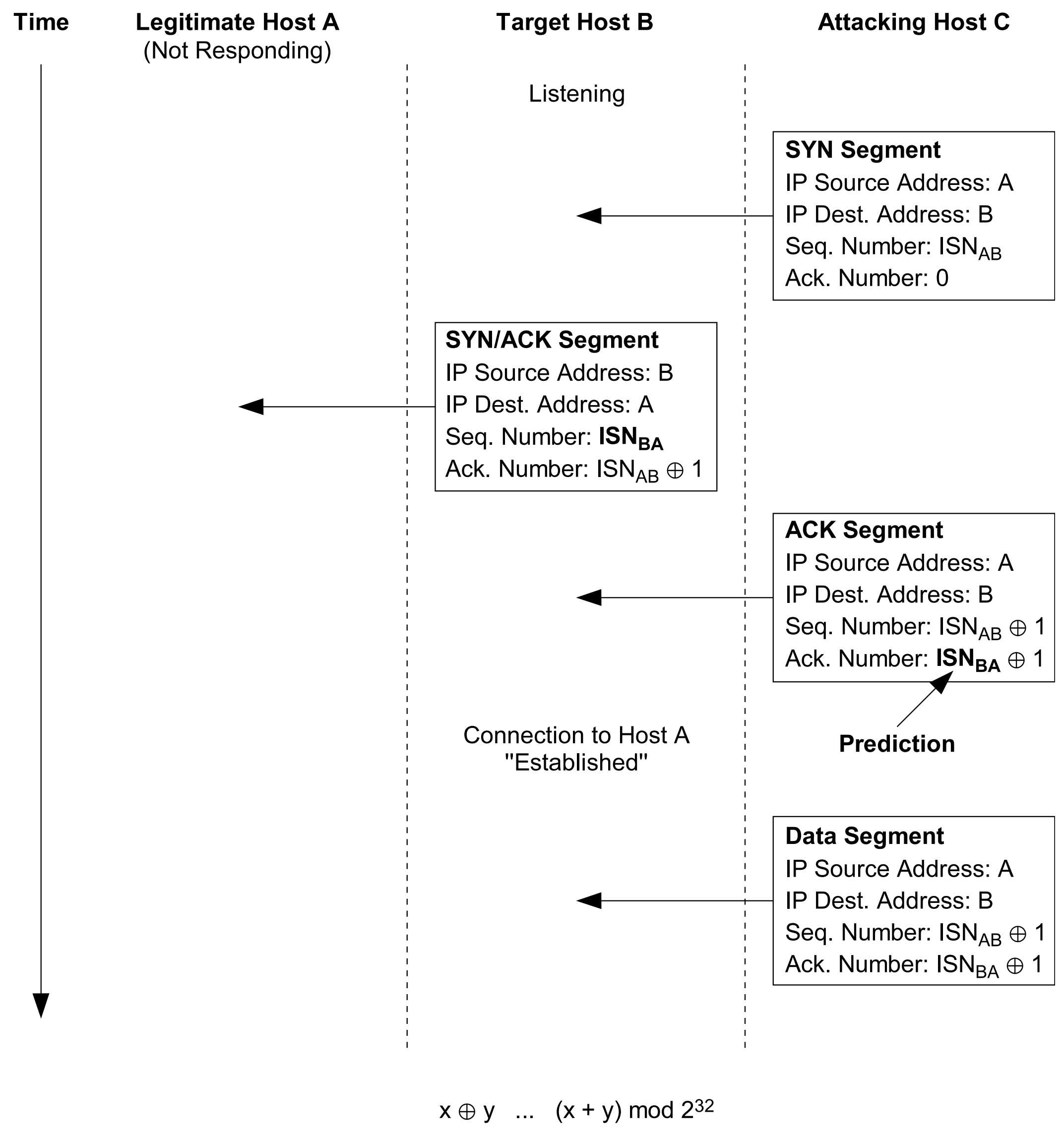

Time
Legitimate Host A (Not Responding)
Target Host B
Attacking Host C
Listening
SYN Segment
IP Source Address: A
IP Dest. Address: B
Seq. Number: ISN_{AB}
Ack. Number: 0
SYN/ACK Segment
IP Source Address: B
IP Dest. Address: A
Seq. Number: $\mathbf{ISN_{BA}}$
Ack. Number: $ISN_{AB} \oplus 1$
ACK Segment
IP Source Address: A
IP Dest. Address: B
Seq. Number: $ISN_{AB} \oplus 1$
Ack. Number: $\mathbf{ISN_{BA}} \oplus 1$
Prediction
Connection to Host A "Established"
Data Segment
IP Source Address: A
IP Dest. Address: B
Seq. Number: $ISN_{AB} \oplus 1$
Ack. Number: $ISN_{BA} \oplus 1$
$x \oplus y$ ... $(x + y) \bmod 2^{32}$

Figure 6.16: TCP/IP Spoofing

ber field is sent to the target. The prediction can be made based on the first ISN received from the SSH server port because the TCP module in the LXRT task advances the ISNs predictably dependent on the progression of time. In contrast, the Linux TCP module of the FTP server does not adhere to such a simple scheme which makes a TCP/IP spoofing attack more difficult. However, selecting the ISNs dependent on the progression of time is in line with the original TCP standards [USC81b, IET89]. As a consequence, simple ISN advancing schemes might appear in practice, e.g., if a special TCP/IP stack for embedded systems is used. After the ACK segment a segment containing the HTTP GET request “`GET /ConfigurationSet.html?Parameter1=on`” is sent in order to activate Parameter1 inside the Web server job. After the Web server presumably has responded with an ACK segment (sent to the legitimate host), finally a RST segment aborting the forged connection is sent. Shortly before the start of the experiment the legitimate host has been disconnected from the local network in order to simulate a non responding host (see above). The sequence of steps (from the SYN segment to the SSH server to the final RST segment) had to be repeated a few times in order to let the ISN prediction succeed. This has been expected because several subsystems between the Web server job's TCP module and the PC add a jitter. The local network has been implemented by an Ethernet switch.

The success of the experiment has been verified by requesting the Configuration page (Figure 6.4) from the Web server displaying that the value of Parameter1 indeed had been changed.

Local Attack: Another experiment has been performed demonstrating how to perpetrate an attack from the local network of the legitimate host, i.e., the host not blocked by the firewall (Figure 6.14). In contrast to Figure 6.14 the attack did not involve the Internet between the two routers. Instead only one router has been in place between the cluster and the local network of legitimate host and attacking host (see Figures 6.17 and 6.12). This attack simply employed the reconfiguration of the attacking host to use the IP address of the legitimate host (via the `ifconfig` command). After the reconfiguration, in spite of the active firewall the attacking host had full access to the cluster as if it were the legitimate host. During the attack the legitimate host has been up and connected to the local network. This local network had been implemented by a switch forwarding packets only to the destination. Although, this provides a higher level of security than a hub forwarding packets to all connected hosts, it could not avert this kind of attack. This attack utilizes the ARP (Address Resolution Protocol) [Plu82] query-reply scheme. ARP is used for the resolution of IP addresses to Ethernet addresses.

External Eavesdropping Attacks

ARP mechanisms have also been utilized in an eavesdropping experiment. As in the local IP spoofing experiment above, the attacking host again has been connected to the local network (Figure 6.17). The local network again has been implemented by a switch forwarding packets only to destination hosts (in contrast to broadcasting packets to all hosts as normally done by hubs). The tool ettercap NG version 0.7.3 [Ett] has been instructed to send forged ARP messages to the

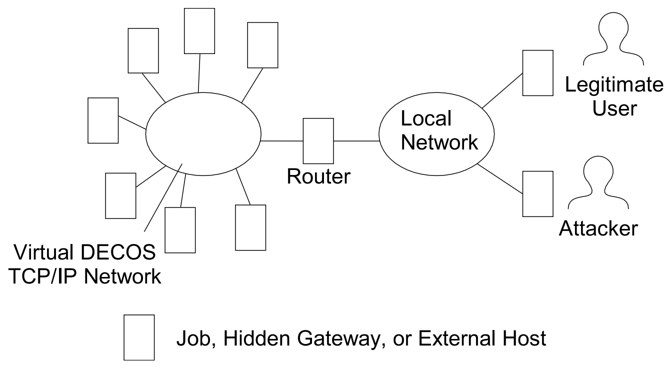

Legitimate User
Local Network
Router
Attacker
Virtual DECOS TCP/IP Network
Job, Hidden Gateway, or External Host

Figure 6.17: Local Attack Scenario in Evaluation

local network in order to redirect traffic between the cluster and the legitimate host through the attacking host. The tool ethereal [Eth] has been used at the attacking host in order to log the packets exchanged between the cluster and the legitimate host. As a consequence, an FTP username and the corresponding password could be logged at the attacking host. This is easily possible due to the fact that FTP usernames and passwords are sent to the network in clear. The experiment has been conducted while the firewall but no IPsec has been active at the security gateway.

External Denial-of-Service Attacks

A further kind of attack is a Denial-of-Service (DoS) attack employing IP spoofing. In the corresponding experiment the attacking host again has been connected to the local network of the legitimate host (Figures 6.17, 6.12) which again has been implemented by a switch. As the TCP spoofing attack described above, in principle this attack can be launched from anywhere on the Internet. However, in this case, the attack might be hampered by ingress and egress filters (as above in the TCP spoofing scenario) or a too low bandwidth between attacker and target. Possibly, the latter hindrance can be eliminated by employing multiple originating attacking hosts, i.e., performing a distributed DoS (DDoS) attack.

In the first part of the experiment, a file containing 10240 random bytes has been transferred via FTP from the legitimate host to the FTP server job (see Figure 6.12) without a DoS attack being perpetrated. Once only the firewall has been active and another time the firewall and IPsec have been active. In the IPsec case, an active tunnel had been established between the legitimate host and the security gateway. In both cases the transfer finished correctly and took approximately 13 to 16 seconds (the duration varied slightly when repeating the transfer).

In the second part of the experiment, the file has been transferred while a

DoS attack has been perpetrated on the cluster-internal virtual TCP/IP network. The DoS attack involved the flooding of the Web server job's TCP/IP stack with TCP SYN segments. The flooding has been accomplished by a Linux shell script repeatedly calling the tool `sendip` [Sen] which can send user defined IP datagrams. In the IP header a forged IP source address not blocked by the firewall has been inserted. Again, once only the firewall has been active and another time the firewall and IPsec have been active. In the former case, the transfer could not be finished successfully. Instead, only a part of the file could be transferred. In a first attempt, the DoS script has been stopped manually after 8 minutes and in a second transfer attempt the DoS script has been stopped manually after 12 minutes. In both cases, the transfer did not continue or finish even after that even though a "ping" to the FTP server job worked. In the scenario with IPsec activated, the transfer finished normally in spite of the attack and took about 13 seconds (as in the above scenario without the attack).

On the one hand, the experiment demonstrates the feasibility of a DoS attack in spite of the firewall and on the other hand it confirms the effectiveness of IPsec virtual private networks in protecting against DoS attacks.

6.3.3 Overview of Vulnerabilities

Table 6.1 presents an overview of the vulnerabilities examined in the vulnerability analysis. Furthermore, the table indicates the relations of the vulnerabilities to the security properties integrity, availability, and confidentiality. A vulnerability assigned to a security property enables a successful attack impacting that property if successfully exploited. The relation references only the immediate relation of a vulnerability to a security property. For example, the vulnerability allowing the identification of the operating system version might not only lead to a successful attack on confidentiality (identification of operating system version), but indirectly might even lead to a successful attack on integrity or availability if the knowledge of the operating system version enables such an attack. This could be the case if vulnerabilities of the version are commonly known.

6.3.4 Discussion of Experimental Evaluation

The security gateway has been found to be very effective in protecting the cluster-internal network from potential attackers on the Internet. The VPN approach offers a high degree of protection from external threats while providing unrestricted services to authorized users. Moreover, it has been demonstrated that a firewall

Vulnerability	Integrity	Availability	Confidentiality
Service Identification (1.(a), 2.(a), 2.(b))			●
Operating System Identification (1.(b))			●
TCP SYN/FIN (1.(c))	●	●	●
TCP RST (1.(d))		●	
ICMP Timestamp (1.(e), 3.(a))			●
Traceroute (1.(f), 2.(c), 3.(b))			●
FTP PORT Command (1.(g))			●
FTP PASV Command (1.(h))		●	
Remote TCP/IP Spoofing Vulnerability	●	●	
Local IP Spoofing Vulnerability	●	●	●
Local Eavesdropping Vulnerability			●
IP Spoofing DoS Vulnerability		●	

Table 6.1: Relations of Vulnerabilities to Security Properties

alone does not offer sufficient protection from attacks. The experiments described above are targeted against a combination of custom-made software (the TCP/IP stacks in LXRT tasks and the Web server) and widely used open source software (the Linux TCP/IP stack, iptables, IPsec, and the FTP server). In many other cases, e.g., in the case of industrial proprietary embedded system software, similar vulnerabilities might be encountered as long as standard communication protocols such as IP, TCP, HTTP, FTP, etc. are employed.

Chapter 7

Conclusion and Future Work

Motivated by advantages expected from incorporating standard communication protocols into embedded systems, this thesis presents an embedded system model providing secure Internet interfaces. In addition, an implementation based on an existing prototype cluster is presented and experimentally evaluated with respect to security.

Motivation and Objective

The expected advantages of employing standard communication protocols are interoperability, ease of training, and the possibility of using COTS components [Kuc98]. On the other hand, standard protocols might introduce security risks due to well-known vulnerabilities [Gar00, HM96, Mar00, KaC00]. As a consequence, the standard interfaces must be secured and most importantly the safety-critical part of an ultra-dependable embedded system has to be protected from malicious activity originating in the Internet.

System Model

The embedded system model is based on the DECOS integrated system architecture [OPT07, DEC]. Services are offered via standard Internet communication protocols such as IP, UDP, TCP, and HTTP. Protection from malicious faults is achieved by physical shielding, a firewall, authentication at network and application level, encryption, job partitioning, job intrusion detection, and subsystem partitioning. Some of these mechanisms protect the system in a layered fashion as the protective layers in the Swiss Cheese Model. As a consequence, the circumvention of one mechanism, e.g., the firewall, does not lead to an intrusion if the next protective layer, e.g., cryptographic authentication by IPsec, blocks the attack.

The concepts of fault-containment region (FCR) and error-containment region (ECR) which have been applied in the literature to non-malicious faults are applied in the system model to various kinds of malicious faults. The use of unified concepts for malicious and non-malicious faults might ease system design and reduce the effort required in order to understand the system and its underlying theories. Similarly, the use of mechanisms such as partitioning and out-of-norm assertions (ONAs) against malicious *and* non-malicious faults might simplify the design.

The model supports the implementation of several kinds of applications demanding for secure remote interfaces. Security, i.e., confidentiality, integrity, and availability, is important in many application scenarios such as remote fault diagnosis or remote model refinement utilizing data collected from the fleet-in-the-field.

Implementation

A DECOS prototype cluster is used in the implementation to demonstrate an exemplary TCP/IP virtual network on top of a TTP/C network and the corresponding security services. IP, UDP, ICMP modules, and in cooperation with the author of [Ben04] an HTTP server module have been implemented. The TCP module implemented by the author of [Ben04] is also used in the implementation to allow communication via standard Internet protocols. A security gateway containing a firewall and an IPsec module has been inserted inside a component interfacing to an Ethernet. The IPsec module provides cryptographic authentication and encryption services.

The implementation follows the system model to the extent possible in the course of this work. Subsystem partitioning at the component level is enforced by the employment of distinct hardware nodes. At the network level, partitioning is achieved by the Ethernet TDMA scheme and the basic connector unit. However, for simplicity the intra-component network is implemented by an Ethernet hub limiting partitioning. At job-granularity partitioning is achieved by various mechanisms: The use of LXRT allows the utilization of Linux memory protection mechanisms. The use of a special scheduler [HPOS05] inhibits temporal fault propagation by deadline violation. At the network-level, the complex connector unit inhibits temporal fault propagation between jobs. However, it remains vulnerable because the interface to the jobs is implemented by shared memory without cryptographic protection.

Results and Evaluation

In order to allow the evaluation of the security gateway, a configuration service via the HTTP server and an FTP server have been installed. Vulnerability scanners have been employed in order to compare weaknesses of the unprotected and the protected system. The results indicate the effectiveness of the security gateway. Moreover, a TCP spoofing, an IP spoofing, an eavesdropping, and a DoS attack have been perpetrated against the system protected by the firewall only. The success of these attacks confirmed that additional protection mechanisms such as authentication are advisable. The DoS attack has been demonstrated to be in vain if both the firewall and IPsec had been activated. The above mentioned weaknesses of the insufficiently protected system have been assigned to the security properties of integrity, availability, and confidentiality. This assignment illustrated that for each security property several vulnerabilities have been found for the insufficiently protected system.

Future Work

The system model could be extended by the specification of configuration interfaces for the security services (IPsec, firewall). The change of passwords or blocked IP addresses must be possible without weakening security. Moreover, a secure service enabling the reset of a forgotten password might be necessary.

Software-update interfaces for the safety-critical subsystem might be a promising research topic due to the possible benefit with respect to safety and economic efficiency. On the one hand, certification of the safety-critical subsystem up to the highest criticality class still would have to be possible and on the other hand, the interfaces would have to be sufficiently secure in order to keep the risk of a malicious software update very low.

The presented system model could be fully implemented and experimentally evaluated. Many embedded systems are physically accessible to unauthorized people, e.g., the embedded system of a car. As a consequence, physical protection could be improved by implementing a component as a system-on-chip as proposed in [KOPS04]. Partitioning at the granularity of both subsystems and jobs should be implemented to the full extent.

Bibliography and Web References

[AAC$^+$03] A. Adelsbach, D. Alessandri, C. Cachin, S. Creese, Y. Deswarte, K. Kursawe, J.-C. Laprie, D. Powell, B. Randell, J. Riordan, P. Ryan, W. Simmonds, R. Stroud, P. Veríssimo, M. Waidner, and A. Wespi. Conceptual Model and Architecture of MAFTIA. DI–FCUL TR–03–01, Department of Informatics, University of Lisbon, February 2003. Project MAFTIA IST–1999–11583 deliverable D21.

[AFTO04] J. Alves-Foss, C. Taylor, and P. Oman. A Multi-layered Approach to Security in High Assurance Systems. In *Proceedings of the 37th Annual Hawaii International Conference on System Sciences, 2004*, January 2004.

[AHS06] André Adelsbach, Ulrich Huber, and Ahmad-Reza Sadeghi. Secure Software Delivery and Installation in Embedded Systems. In Kerstin Lemke, Christof Paar, and Marko Wolf, editors, *Embedded Security in Cars: Securing Current and Future Automotive IT Applications*, pages 27–49. Springer-Verlag, 2006.

[ALR01] A. Avizienis, J.-C. Laprie, and B. Randell. Fundamental Concepts of Dependability. Technical report, University of Newcastle, Department of Computing Science, 2001.

[ALRL04] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr. Basic Concepts and Taxonomy of Dependable and Secure Computing. In *IEEE Transactions on Dependable and Secure Computing*, volume 1 (1), pages 11–33, 2004.

[AM04] S.V. Amari and L. McLaughlin. Optimal Design of a Condition-Based Maintenance Model. In *Reliability and Maintainability, 2004 Annual Symposium - RAMS*, volume 3, pages 528–533, 2004.

[ARI] *ARINC Standards – 600 Series.* Web site. 2007-04-19. URL: `https://www.arinc.com/cf/store/catalog.cfm?prod_group_id=1&category_group_id=3`.

[Bak02] Fred Baker. *Internet Reliability under Stress*, 2002. Presentation slides. Source-URL: `http://soft.com/QualWeek/QW2002/downloads/1G1.presentation.pdf` (2007-03-09).

[Bau01] R.C. Baumann. Soft Errors in Advanced Semiconductor Devices—Part I: The Three Radiation Sources. In *IEEE Transactions on Device and Materials Reliability*, volume 1 (1), pages 17–22, March 2001.

[Ben02] Jeremy Bentham. *TCP/IP Lean: Web Servers for Embedded Systems.* CMP Books, second edition, 2002.

[Ben04] Roman Benesch. *TCP für die Time-Triggered Architecture.* Diploma thesis, University of Technology Vienna, Vienna, Austria, 2004.

[BM01] Rebecca Bace and Peter Mell. Intrusion Detection Systems. Special Publication SP 800-31, National Institute of Standards and Technology, Computer Security Resource Center (CSRC), November 2001. Source-URL: `http://csrc.nist.gov/publications/nistpubs/800-31/sp800-31.pdf` (2006-10-08).

[Bra96] S. Bradner. *The Internet Standards Process – Revision 3.* Network Working Group, October 1996. RFC 2026, Source-URL: `http://www.rfc-editor.org` (2007-02-12).

[CB05] A. Creery and E.J. Byres. Industrial Cybersecurity for Power System and SCADA Networks. In *Industry Applications Society 52nd Annual Petroleum and Chemical Industry Conference, 2005*, pages 303–309, September 2005.

[CC04] Marco Carugi and Jeremy De Clercq. Virtual Private Network Services: Scenarios, Requirements and Architectural Constructs from a Standardization Perspective. *IEEE Communications Magazine*, pages 116–122, June 2004.

[CER] *Problems With The FTP PORT Command.* Web site. 2006-04-20. URL: `http://www.cert.org/tech_tips/ftp_port_attacks.html`.

[CGI] *Common Gateway Interface (CGI).* Web site. 2006-04-02. URL: http://hoohoo.ncsa.uiuc.edu/cgi/overview.html.

[CGPS03] F. Callegati, R. Gori, P. Presepi, and M. Sacchetti. Implementation of a Micro Web Server for Peer-to-Peer Applications. In G. Moro and M. Koubarakis, editors, *Lecture Notes in Computer Science*, volume 2530, pages 164–169. Springer-Verlag, Heidelberg, 2003.

[CHB02] M. Cilia, P. Hasselmeyer, and A.P. Buchmann. Profiling and Internet Connectivity in Automotive Environments. In *Proceedings of 28th International Conference on Very Large Data Bases, 2002, Hong Kong, China. VLDB 2002*, pages 1071–1074. Morgan Kaufmann, 2002. Source-URL: http://www.vldb.org/conf/2002/S33P08.pdf (2007-05-09).

[Cis] Cisco Systems, Inc. *Internetworking Technology Handbook.* Source-URL: http://www.cisco.com (2004-08-21).

[CO02] R. Chakravorty and H. Ottevanger. Architecture and Implementation of a Remote Management Framework for Dynamically Reconfigurable Devices. In *10th IEEE International Conference on Networks, 2002. ICON 2002.*, pages 375–380, 2002.

[Con02] C. Constantinescu. Impact of Deep Submicron Technology on Dependability of VLSI Circuits. In *Proceedings of the International Conference on Dependable Systems and Networks*, pages 205–209, June 2002.

[CVEa] *Common Vulnerabilities and Exposures — CVE-1999-0017.* Web site. 2006-04-04. URL: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-1999-0017.

[CVEb] *Common Vulnerabilities and Exposures — CVE-1999-0079.* Web site. 2006-04-04. URL: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-1999-0079.

[CVEc] *Common Vulnerabilities and Exposures — CVE-1999-0524.* Web site. 2006-04-03. URL: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-1999-0524.

[DA99] T. Dierks and C. Allen. *The TLS Protocol Version 1.0.* Network Working Group, January 1999. RFC 2246, Source-URL: `http://www.rfc-editor.org` (2007-03-14).

[DEC] *DECOS (Dependable Embedded Components and Systems).* Web site. 2005-11-21. URL: `http://www.decos.at`.

[DFF$^+$88] Y. Deswarte, J.-C. Fabre, J.-M. Fray, D. Powell, and P.-G. Ranea. SATURNE: A Distributed Computing System Which Tolerates Faults and Intrusions. In *Proceedings of the Workshop on the Future Trends of Distributed Computing Systems in the 1990s, 1988*, pages 329–338, September 1988.

[DFMP98] Elmar Dilger, Thomas Führer, Bernd Müller, and Stefan Poledna. The X-By-Wire Concept: Time-Triggered Information Exchange and Fail Silence Support by new System Services. Research Report 7/1998, University of Technology Vienna, Institute of Computer Engineering, 1998.

[DH95] S. Deering and R. Hinden. *Internet Protocol, Version 6 (IPv6) Specification.* Network Working Group, December 1995. RFC 1883, Source-URL: `http://www.rfc-editor.org` (2007-02-12).

[DH98] S. Deering and R. Hinden. *Internet Protocol, Version 6 (IPv6) Specification.* Network Working Group, December 1998. RFC 2460, Source-URL: `http://www.rfc-editor.org` (2007-02-12).

[DKS89] A. Demers, S. Keshav, and S. Shenker. Analysis and Simulation of a Fair Queueing Algorithm. In *Symposium Proceedings on Communications Architectures & Protocols*, pages 1–12. ACM Press, 1989.

[DNvHC05] D. Dzung, M. Naedele, T.P. von Hoff, and M. Crevatin. Security for Industrial Communication Systems. *Proceedings of the IEEE*, 93 (6):1152–1177, June 2005.

[Dod01] D.S. Dodge. Gateways - 101. In *Military Communications Conference, 2001. MILCOM 2001. Communications for Network-Centric Operations: Creating the Information Force*, volume 1, pages 532–538. IEEE, October 2001.

[DR01a] J. Dobson and B. Randell. Introduction to “Building Reliable Secure Computing Systems out of Unreliable Insecure Components”. In

Proceedings 17th Annual Computer Security Applications Conference, 2001. ACSAC 2001, pages 162–163, December 2001.

[DR01b] J.E. Dobson and B. Randell. Building Reliable Secure Computing Systems out of Unreliable Insecure Components. In *Proceedings 17th Annual Computer Security Applications Conference, 2001. ACSAC 2001*, pages 164–173, December 2001.

[Dun01] Adam Dunkels. Minimal TCP/IP implementation with proxy support. Technical report, Swedish Institute of Computer Science, 2001.

[Ele] Electronic Design. *Control Your Failures in Time and Keep Customers Happy.* Web site. 2001-10-15. URL: `http://www.elecdesign.com/Articles/Index.cfm?ArticleID=3619` (2005-07-06).

[Emb] Embedded Star. *Fujitsu Demonstrates MOST Multimedia Car Infotainment System.* Web site. 2004-11-11. URL: `http://www.embeddedstar.com/press/content/2004/11/embedded17136.html` (2005-07-06).

[Eth] *Ethereal: A Network Protocol Analyzer.* Web site. 2007-02-20. URL: `http://www.ethereal.com`.

[Ett] *Ettercap NG.* Web site. 2007-02-20. URL: `http://ettercap.sourceforge.net`.

[FGM+99] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. *Hypertext Transfer Protocol — HTTP/1.1*, June 1999. RFC 2616, Source-URL: `http://www.rfc-editor.org` (2004-03-07).

[Fur03] Frank J. Furrer. *Industrieautomation mit Ethernet-TCP/IP und Web-Technologie.* Hüthig, third edition, 2003.

[Gar00] L. Garber. Denial-of-Service Attacks Rip the Internet. *IEEE Computer*, 33 (4):12–17, 2000.

[GIJ+03] M.-C. Gaudel, V. Issarny, C. Jones, H. Kopetz, E. Marsden, N. Moffat, M. Paulitsch, D. Powell, B. Randell, A. Romanovsky, R. Stroud, and F. Taiani. Final Version of DSoS Conceptual Model (DSoS Project deliverable CSDA1). Technical Report CS-TR-782, University of Newcastle upon Tyne, 2003.

[GMF$^+$05] Vipul Gupta, Matthew Millard, Stephen Fung, Yu Zhu, Nils Gura, Hans Eberle, and Sheueling Chang Shantz. Sizzle: A Standards-based end-to-end Security Architecture for the Embedded Internet. In *Third IEEE International Conference on Pervasive Computing and Communications, 2005. PerCom 2005*, pages 247–256, March 2005.

[GW96] Ian Goldberg and David Wagner. Randomness and the Netscape Browser. *Dr. Dobb's Journal*, January 1996. Source-URL: `http://www.ddj.com/documents/s=965/ddj9601h/` (2006-04-04).

[Her99] Eli Herscovitz. Secure Virtual Private Networks: The Future of Data Communications. *International Journal of Network Management*, 9 (4):213–220, 1999.

[Hex03] René Hexel. FITS — A Fault Injection Architecture for Time-Triggered Systems. In *Proceedings of the Twenty-Sixth Australasian Computer Science Conference*, volume 16, pages 333–338. Australian Computer Society, Inc., 2003.

[HL00] M.G. Hill and T.W. Lake. Non-Interference Analysis for Mixed Criticality Code in Avionics Systems. In *Proceedings of the Fifteenth IEEE International Conference on Automated Software Engineering (ASE 2000)*, pages 257–260, 2000.

[HM96] N.E. Hastings and P.A. McLean. TCP/IP Spoofing Fundamentals. In *Conference Proceedings of the 1996 IEEE Fifteenth Annual International Phoenix Conference on Computers and Communications, 1996.*, pages 218–224, March 1996.

[HOB$^+$06] J. Holleman, B. Otis, S. Bridges, A. Mitros, and C. Diorio. A 2.92μW Hardware Random Number Generator. In *Proceedings of the 32nd European Solid-State Circuits Conference, 2006. ESSCIRC 2006*, pages 134–137, September 2006.

[HPOS05] B. Huber, P. Peti, R. Obermaisser, and C. El Salloum. Using RTAI/LXRT for Partitioning in a Prototype Implementation of the DECOS Architecture. In *Proceedings of the Third International Workshop on Intelligent Solutions in Embedded Systems (WISES 2005)*, 2005.

[HS99] Michael Howard and Christopher S. Sontag. Bringing the Internet to All Electronic Devices. In *Proceedings of the Embedded Systems Workshop*. USENIX, 1999.

[idQ] *id Quantique: Quantis - Quantum Random Number Generators*. Web site. 2007-04-20. URL: `http://www.idquantique.com/products/quantis.htm`.

[IET89] Internet Engineering Task Force. *Requirements for Internet Hosts — Communication Layers*, October 1989. RFC 1122, Source-URL: `http://www.rfc-editor.org` (2004-03-06).

[Int] Internet Systems Consortium, Inc. *Internet Domain Survey*. Web site. 2006-12-29. URL: `http://www.isc.org`.

[IPR98] R. Itschner, C. Pommerell, and M. Rutishauser. GLASS: Remote Monitoring of Embedded Systems in Power Engineering. *IEEE Internet Computing*, 2 (3):46–52, 1998.

[IPs] *IPsec-Tools Homepage*. Web site. 2006-03-29. URL: `http://ipsec-tools.sourceforge.net`.

[JBB92] V. Jacobson, R. Braden, and D. Borman. *TCP Extensions for High Performance*. Network Working Group, May 1992. RFC 1323, Source-URL: `http://www.rfc-editor.org` (2007-02-13).

[JCH00] Hong-Taek Ju, Mi-Joung Choi, and James W. Hong. An Efficient and Lightweight Embedded Web Server for Web-Based Network Element Management. *International Journal of Network Management*, 10 (5):261–275, 2000.

[Jon06] E. Jonsson. Towards an Integrated Conceptual Model of Security and Dependability. In *The First International Conference on Availability, Reliability and Security, 2006. ARES 2006*, pages 646–653, April 2006.

[JSJF98] A. Jameel, M. Stuempfle, D. Jiang, and A. Fuchs. Web on Wheels: Toward Internet-Enabled Cars. *Computer*, 31 (1):69–76, January 1998.

[KaC00] Othmar Kyas and Markus a Campo. *IT Crackdown*. MITP-Verlag, third edition, 2000.

[KB03] H. Kopetz and G. Bauer. The Time-Triggered Architecture. In *Proceedings of the IEEE*, volume 91 (1), pages 112–126, 2003.

[KDH+01] M. Klausner, A. Dietrich, J.-P. Hathout, A. Springer, B. Seubert, and P. Stumpp. Vehicle Data Management System with Remote Access to Electronic Control Unit-Internal States. In *International Conference on Advanced Driver Assistance Systems, 2001. ADAS.*, pages 68–72, 2001.

[Ken05] S. Kent. *IP Authentication Header.* Network Working Group, December 2005. RFC 4302, Source-URL: `http://www.rfc-editor.org` (2007-04-03).

[ker] *A Strong Random Number Generator.* Linux kernel source code. 2006-05-16. URL: `http://www.kernel.org`, file `linux-2.6.9/drivers/char/random.c` in `linux-2.6.9.tar.bz2`.

[KG94] H. Kopetz and G. Grünsteidl. TTP — A Protocol for Fault-Tolerant Real-Time Systems. *IEEE Computer*, 27 (1):14–23, 1994.

[KHE00] H. Kopetz, M. Holzmann, and W. Elmenreich. A Universal Smart Transducer Interface: TTP/A. In *Proceedings Third IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC 2000)*, pages 16–23, 2000.

[KKM+98] H. Kopetz, M. Kucera, D. Millinger, C. Ebner, and I. Smaili. Interfacing Time-Triggered Embedded Systems to the INTERNET. Research Report 10/1998, University of Technology Vienna, Institute of Computer Engineering, Vienna, Austria, 1998.

[KLM+04] P. Kocher, R. Lee, G. McGraw, A. Raghunathan, and S. Ravi. Security as a New Dimension in Embedded System Design. In *Proceedings. 41st Design Automation Conference, 2004*, pages 753–760, 2004.

[Koo04] P. Koopman. Embedded System Security. *Computer*, 37 (7):95–97, July 2004.

[Kop97] H. Kopetz. *Real-Time Systems: Design Principles for Distributed Embedded Applications.* Kluwer Academic Publishers, 1997.

[Kop00] H. Kopetz. Software Engineering for Real-Time: A Roadmap. In *Proceedings of the Conference on the Future of Software Engineering*, pages 201–211, New York, NY, USA, 2000. ACM Press.

[Kop03] H. Kopetz. Fault Containment and Error Detection in the Time-Triggered Architecture. In *Sixth International Symposium on Autonomous Decentralized Systems, 2003 (ISADS 2003)*, pages 139–146, April 2003.

[Kop04] H. Kopetz. The Fault Hypothesis for the Time-Triggered Architecture. In René Jacquart, editor, *Building the Information Society*, volume 156 of *IFIP International Federation for Information Processing*, pages 221–233. Kluwer Academic Publishers, 2004.

[Kop06] H. Kopetz. *Real-Time Systems – System Design*, 2006. Lecture slides from the lecture "Real-Time Systems", Source-URL: `http://ti.tuwien.ac.at/rts/teaching/courses/ezs/pdf-slides/7_system_design.pdf` (2007-02-19).

[KOPS04] H. Kopetz, R. Obermaisser, P. Peti, and N. Suri. From a Federated to an Integrated Architecture for Dependable Real-Time Embedded Systems. Research Report 22/2004, University of Technology Vienna, Institute of Computer Engineering, Vienna, Austria, 2004.

[KS03] H. Kopetz and N. Suri. Compositional Design of RT Systems: A Conceptual Basis for Specification of Linking Interfaces. In *Sixth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC'03)*, pages 51–60, May 2003.

[Kuc98] Markus Kucera. *On the Cooperation between Time-Triggered Real-Time Systems and Event-Triggered Internet-Based Systems.* PhD thesis, University of Technology Vienna, Institute of Computer Engineering, 1998.

[KV02] R. A. Kemmerer and G. Vigna. Intrusion Detection: A Brief History and Overview. *Computer*, 35 (4):27–30, 2002.

[Lem06] Kerstin Lemke. Embedded Security: Physical Protection against Tampering Attacks. In Kerstin Lemke, Christof Paar, and Marko Wolf, editors, *Embedded Security in Cars: Securing Current and Future Automotive IT Applications*, pages 207–217. Springer-Verlag, 2006.

[LGK98] T. Lumpp, G. Gruhler, and W. Kuchlin. Virtual Java Devices. Integration of Fieldbus Based Systems in the Internet. In *Industrial*

Electronics Society, 1998. IECON '98. Proceedings of the 24th Annual Conference of the IEEE, volume 1, pages 176–181, 1998.

[LH02] G. Leen and D. Heffernan. Expanding Automotive Electronic Systems. *Computer*, 35 (1):88–93, January 2002.

[Mar00] Kai Martius. *Sicherheitsmanagement in TCP/IP-Netzen.* Vieweg, 2000.

[MB03] C. Maihofer and M. Bechler. Design Alternatives for IP in Vehicles. In *The 57th IEEE Semiannual Vehicular Technology Conference, 2003. VTC 2003-Spring*, volume 3, pages 1783–1787, April 2003.

[McK90] Paul E. McKenney. Stochastic Fairness Queueing. In *Ninth Annual Joint Conference of the IEEE Computer and Communication Societies. 'The Multiple Facets of Integration'. INFOCOM '90.*, volume 2, pages 733–740, June 1990.

[MDS02] T.S. Messerges, E.A. Dabbish, and R.H. Sloan. Examining Smart-Card Security under the Threat of Power Analysis Attacks. In *IEEE Transactions on Computers*, volume 51 (5), pages 541–552, May 2002.

[Mea94] Catherine Meadows. The Need for a Failure Model for Security. Technical report, Naval Research Laboratory, Center for High Assurance Computer Systems, Washington, D.C., 1994. Source-URL: `http://chacs.nrl.navy.mil/publications/CHACS/1994/1994meadows-dcca4A.pdf` (2007-05-10).

[Moc87a] P. Mockapetris. *Domain Names — Concepts and Facilities*, November 1987. RFC 1034, Source-URL: `http://www.rfc-editor.org` (2004-03-07).

[Moc87b] P. Mockapetris. *Domain Names — Implementation and Specification*, November 1987. RFC 1035, Source-URL: `http://www.rfc-editor.org` (2004-03-07).

[MOS] MOST Cooperation. *MOST Technology.* Web site. 2005-02-05. URL: `http://www.mostnet.de`.

[Nag87] J. Nagle. On Packet Switches with Infinite Storage. In *IEEE Transactions on Communications*, volume 35 (4), pages 435–438, April 1987.

[Nes] *Nessus Vulnerability Scanner*. Web site. 2006-03-30. URL: `http://www.nessus.org`.

[net] *iptables*. Web site. 2006-03-29. URL: `http://www.netfilter.org/projects/iptables/index.html`.

[Nex02] *ET & TT Requirements Document*, 2002. Project NEXT TTA IST–2001–32111 deliverable D2.1.

[Nex03a] *Conceptual Design Document*, 2003. Project NEXT TTA IST–2001–32111 deliverable D2.2.

[Nex03b] *Event-Triggered Services Prototype Implementation*, 2003. Project NEXT TTA IST–2001–32111 deliverable D2.3.

[Nex03c] *Validation Report Document*, 2003. Project NEXT TTA IST–2001–32111 deliverable D2.5.

[NFW01] T. Nieva, A. Fabri, and A. Wegmann. Remote Monitoring of Railway Equipment Using Internet Technologies. Technical report, Swiss Federal Institute of Technology Lausanne, April 2001.

[NM96] D. Naccache and D. M'Raïhi. Cryptographic Smart Cards. *IEEE Micro*, 16 (3):14, 16–24, June 1996.

[NVD] *National Vulnerability Database — CVE-2004-0230*. Web site. 2007-02-11. URL: `http://nvd.nist.gov/nvd.cfm?cvename=CVE-2004-0230`.

[OB06] J. Ott and C. Bormann. *Protocol Design – Security 2: Protocol Design Techniques*, 2006. Lecture slides from the course "Protocol Design", Source-URL: `http://www.netlab.tkk.fi/~jo/teaching/pd/slides/7-security.pdf` (2007-02-19).

[Obe02] Roman Obermaisser. CAN Emulation in a Time-Triggered Environment. In *Proceedings of the 2002 IEEE International Symposium on Industrial Electronics (ISIE 2002)*, volume 1, pages 270–275, 2002.

[OP06] R. Obermaisser and P. Peti. A Fault Hypothesis for Integrated Architectures. In *International Workshop on Intelligent Solutions in Embedded Systems, 2006. WISES '06*, pages 1–18, 2006.

[OPK05] R. Obermaisser, P. Peti, and H. Kopetz. Virtual Networks in an Integrated Time-Triggered Architecture. In *10th IEEE International Workshop on Object-Oriented Real-Time Dependable Systems, 2005. WORDS 2005*, pages 241–253, February 2005.

[Opp00] Rolf Oppliger. *Security Technologies for the World Wide Web.* Artech House, Inc., 2000.

[OPT07] Roman Obermaisser, Philipp Peti, and Fulvio Tagliabo. An Integrated Architecture for Future Car Generations. *Real-Time Systems*, 36 (1–2):101–133, July 2007. Springer Netherlands.

[Pfe01] Olaf Pfeiffer. Embedded Internetworking with 8- and 16-bit Microcontrollers. In *Proceedings of the Embedded Systems Conference West 2001*, San Francisco, 2001.

[Pis02] A. Piskozub. Denial of Service and Distributed Denial of Service Attacks. In *Proceedings of the International Conference on Modern Problems of Radio Engineering, Telecommunications and Computer Science*, pages 303–304, 2002.

[Plu82] David C. Plummer. *An Ethernet Address Resolution Protocol*, November 1982. RFC 826, Source-URL: `http://www.rfc-editor.org` (2004-03-07).

[Poh01] Norbert Pohlmann. *Firewall-Systeme.* MITP-Verlag, fourth edition, 2001.

[POK05] P. Peti, R. Obermaisser, and H. Kopetz. Out-of-Norm Assertions [Diagnostic Mechanism]. In *11th IEEE Real Time and Embedded Technology and Applications Symposium, 2005. RTAS 2005.*, pages 280–291, March 2005.

[Pos80] J. Postel. *User Datagram Protocol*, August 1980. RFC 768, Source-URL: `http://www.rfc-editor.org` (2004-03-07).

[Pos81] J. Postel. *Internet Control Message Protocol*, September 1981. RFC 792, Source-URL: `http://www.rfc-editor.org` (2004-03-07).

[Pow92] D. Powell. Failure Mode Assumptions and Assumption Coverage. In *Twenty-Second International Symposium on Fault-Tolerant Computing (FTCS-22)*, pages 386–395. IEEE, 1992.

[PP03] Charles P. Pfleeger and Shari Lawrence Pfleeger. *Security in Computing.* Prentice Hall Professional Technical Reference, Upper Saddle River, New Jersey 07458, third edition, 2003.

[Pus06] Peter Puschner. *Does the TTA Support Security?*, 2006. Presentation Slides – Panel "Towards Developing a Secure Dependable System: Development, Issues, and Challenges". Ninth IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing, 2006. ISORC 2006.

[PX05] Joshua Pauli and Dianxiang Xu. Threat-Driven Architectural Design of Secure Information Systems. Technical report, North Dakota State University, Department of Computer Science, Fargo, North Dakota, 2005. Source-URL: `http://cs.ndsu.edu/~dxu/publications/pauli-xu-ICEIS05.pdf` (2007-04-12).

[Rea00] James Reason. Human Error: Models and Management. *BMJ (British Medical Journal)*, 320:768–770, 2000.

[Ros90] M.T. Rose. Transition and Coexistence Strategies for TCP/IP to OSI. *IEEE Journal on Selected Areas in Communications*, 8 (1):57–66, January 1990.

[RPH06] Maxim Raya, Panos Papadimitratos, and Jean-Pierre Hubaux. Securing Vehicular Communications. *IEEE Wireless Communications*, 13 (5):8–15, October 2006.

[RRKH04] Srivaths Ravi, Anand Raghunathan, Paul Kocher, and Sunil Hattangady. Security in Embedded Systems: Design Challenges. In *ACM Transactions on Embedded Computing Systems (TECS)*, volume 3 (3), pages 461–491, 2004.

[RTA] *RTAI.* Web site. 2006-03-28. URL: `http://www.rtai.org`.

[Rus99] John Rushby. Partitioning in Avionics Architectures: Requirements, Mechanisms, and Assurance. Technical Report CR–1999–209347, NASA, June 1999.

[SAR] *SARA.* Web site. 2006-03-30. URL: `http://www-arc.com/sara`.

[Sch96] W. Schulz. Traffic Management Improvement by Integrating Modern Communication Systems. *IEEE Communications Magazine*, 34 (10):56–60, October 1996.

[SD02] Joseph S. Sherif and Tommy G. Dearmond. Intrusion Detection: Systems and Models. In *Proceedings of the Eleventh IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, 2002. WET ICE 2002.*, pages 115–133, 2002.

[Sen] *SendIP.* Web site. 2007-02-21. URL: `http://www.earth.li/projectpurple/progs/sendip.html`.

[SGR02] Keith Strassberg, Richard Gondek, and Gary Rollie. *Firewalls: The Complete Reference.* McGraw-Hill/Osborne, 2002.

[Sim94] *The Point-to-Point Protocol (PPP)*, July 1994. W. Simpson, editor, RFC 1661, Source-URL: `http://www.rfc-editor.org` (2004-08-24).

[SL01] S.R. Subramanya and N. Lakshminarasimhan. Computer Viruses. *IEEE Potentials*, 20 (4):16–19, 2001.

[SLG$^+$01] E. Solvang, L. Lundgaard, B. Gustavsen, A. Eggen, and S. Fretheim. Risk Management: Cost Minimization Using Condition-Based Maintenance. In *16th International Conference and Exhibition on Electricity Distribution, 2001 (IEE Conf. Publ No. 482)*, volume 3, page 6 pp., 2001.

[SLP06] Kai Schramm, Kerstin Lemke, and Christof Paar. Embedded Cryptography: Side Channel Attacks. In Kerstin Lemke, Christof Paar, and Marko Wolf, editors, *Embedded Security in Cars: Securing Current and Future Automotive IT Applications*, pages 187–206. Springer-Verlag, 2006.

[Soe] *Soekris Engineering, Inc.* Web site. 2006-03-23. URL: `http://www.soekris.com`.

[SS01] C. Schwaiger and T. Sauter. A Secure Architecture for Fieldbus/Internet Gateways. In *8th IEEE International Conference on Emerging Technologies and Factory Automation, 2001.*, volume 1, pages 279–285, 2001.

[SS02] C. Schwaiger and T. Sauter. Security Strategies for Field Area Networks. In *28th Annual Conference of the Industrial Electronics Society, IEEE 2002 (IECON 02)*, volume 4, pages 2915–2920, 2002.

[Sta98] William Stallings. *Operating Systems: Internals and Design Principles*. Prentice-Hall International, third edition, 1998.

[Ste03] Andreas Steffen. Secure Communications in Embedded Systems. Technical report, Hochschule für Technik Rapperswil, Institut für Internet-Technologien und -Anwendungen, Rapperswil, Switzerland, 2003. Source-URL: `http://security.hsr.ch/docs/embedded_security.pdf` (2007-03-21).

[Ste04] B. Stephens. Security Architecture for Aeronautical Networks. In *The 23rd Digital Avionics Systems Conference, 2004. DASC 04*, volume 2, pages 8.E.2–1–8.E.2–19, October 2004.

[SV96] M. Shreedhar and G. Varghese. Efficient Fair Queuing Using Deficit Round-Robin. In *IEEE/ACM Transactions on Networking*, volume 4 (3), pages 375–385, June 1996.

[SW01] M. Sujecka and B. Wiszniewski. Remote Debugging of CORBA Objects. In *Proceedings of the Ninth Euromicro Workshop on Parallel and Distributed Processing, 2001*, pages 396–401, 2001.

[TAP06] N. Thanthry, M.S. Ali, and R. Pendse. Security, Internet Connectivity and Aircraft Data Networks. *IEEE Aerospace and Electronic Systems Magazine*, 21 (11):3–7, November 2006.

[TTT] *TTTech*. Web site. 2006-03-22. URL: `http://www.tttech.com`.

[TTT03] TTTech Computertechnik AG, Schönbrunner Straße 7, 1040 Vienna, Austria. *TTP Monitoringnode — A TTP Development Board for the Time-Triggered Architecture based on the TTP Chip C2*, December 2003. Source-URL: `http://www.tttech.com` (2006-03-22).

[TUN] *Universal TUN/TAP Driver*. Web site. 2006-03-29. URL: `http://vtun.sourceforge.net/tun/index.html`.

[TVR+99] W. Townsley, A. Valencia, A. Rubens, G. Pall, G. Zorn, and B. Palter. *Layer Two Tunneling Protocol "L2TP"*, August 1999. RFC 2661, Source-URL: `http://www.rfc-editor.org` (2004-08-21).

[U.S99] U.S. Department of Commerce/National Institute of Standards and Technology. *Data Encryption Standard (DES)*, 1999. FIPS PUB 46-3, Source-URL: `http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf` (2007-03-15).

[USC] *US-CERT Vulnerability Notes Database — Vulnerability Note VU#464113*. Web site. 2006-04-04. URL: `http://www.kb.cert.org/vuls/id/464113`.

[USC81a] University of Southern California, Information Sciences Institute. *Internet Protocol*, September 1981. RFC 791, Source-URL: `http://www.rfc-editor.org` (2004-03-07).

[USC81b] University of Southern California, Information Sciences Institute. *Transmission Control Protocol*, September 1981. RFC 793, Source-URL: `http://www.rfc-editor.org` (2004-03-07).

[vBMM90] G. v. Bochmann and P. Mondain-Monval. Design Principles for Communication Gateways. *IEEE Journal on Selected Areas in Communications*, 8 (1):12–21, January 1990.

[Ven01] R. Venkateswaran. Virtual Private Networks. *IEEE Potentials*, 20 (1):11–15, 2001.

[VNC03] Paulo Veríssimo, Nuno Ferreira Neves, and Miguel Correia. Intrusion Tolerant Architectures: Concepts and Design. DI/FCUL TR 03–05, Department of Informatics, University of Lisbon, April 2003. Technical Report, Source-URL: `http://www.di.fc.ul.pt/sobre/?reports#2003` (2006-10-08).

[VP03] Marcus Venzke and Stefan Pitzek. Accessing Fieldbus Systems via Web Services. In *Proceedings of the First Workshop on Intelligent Solutions in Embedded Systems (WISES 2003)*, 2003.

[Was06] Armin Wasicek. Security Considerations in Embedded Systems. Research Report 108/2006, University of Technology Vienna, Institute of Computer Engineering, Vienna, Austria, 2006.

[Wat04a] Paul A. Watson. *Slipping in the Window: TCP Reset Attacks*, 2004. CanSecWest Conference. Source-URL: `http://www.cansecwest.com/csw04archive.html` (2007-02-11).

[Wat04b] Paul A. Watson. *Slipping in the Window: TCP Reset Attacks – Presentation Slides*, 2004. CanSecWest Conference. Source-URL: `http://www.cansecwest.com/csw04archive.html` (2007-02-11).

[WD03] C.A. Wargo and C. Dhas. Security Considerations for the e-Enabled Aircraft. In *2003 IEEE Aerospace Conference*, volume 4, pages 4_1533–4_1550, March 2003.

[WE07] Armin Wasicek and Wilfried Elmenreich. Internet Firewalls in the DECOS System-on-a-Chip Architecture. Research Report 26/2007, University of Technology Vienna, Institute of Computer Engineering, Vienna, Austria, 2007.

[Wei03] C. Weissman. MLS-PCA: A High Assurance Security Architecture for Future Avionics. In *19th Annual Computer Security Applications Conference, 2003. Proceedings*, pages 2–12, 2003.

[Zal02] Robert Zalenski. Firewall Technologies. *IEEE Potentials*, 21 (1):24–29, 2002.

[Zim80] Hubert Zimmermann. OSI Reference Model—The ISO Model of Architecture for Open Systems Interconnection. In *IEEE Transactions on Communications*, volume 28 (4), pages 425–432, April 1980.