The Maftia Architecture: Services and Middleware for Intrusion Tolerance

Paulo Esteves Veríssimo

University of Lisboa

The MAFTIA Consortium

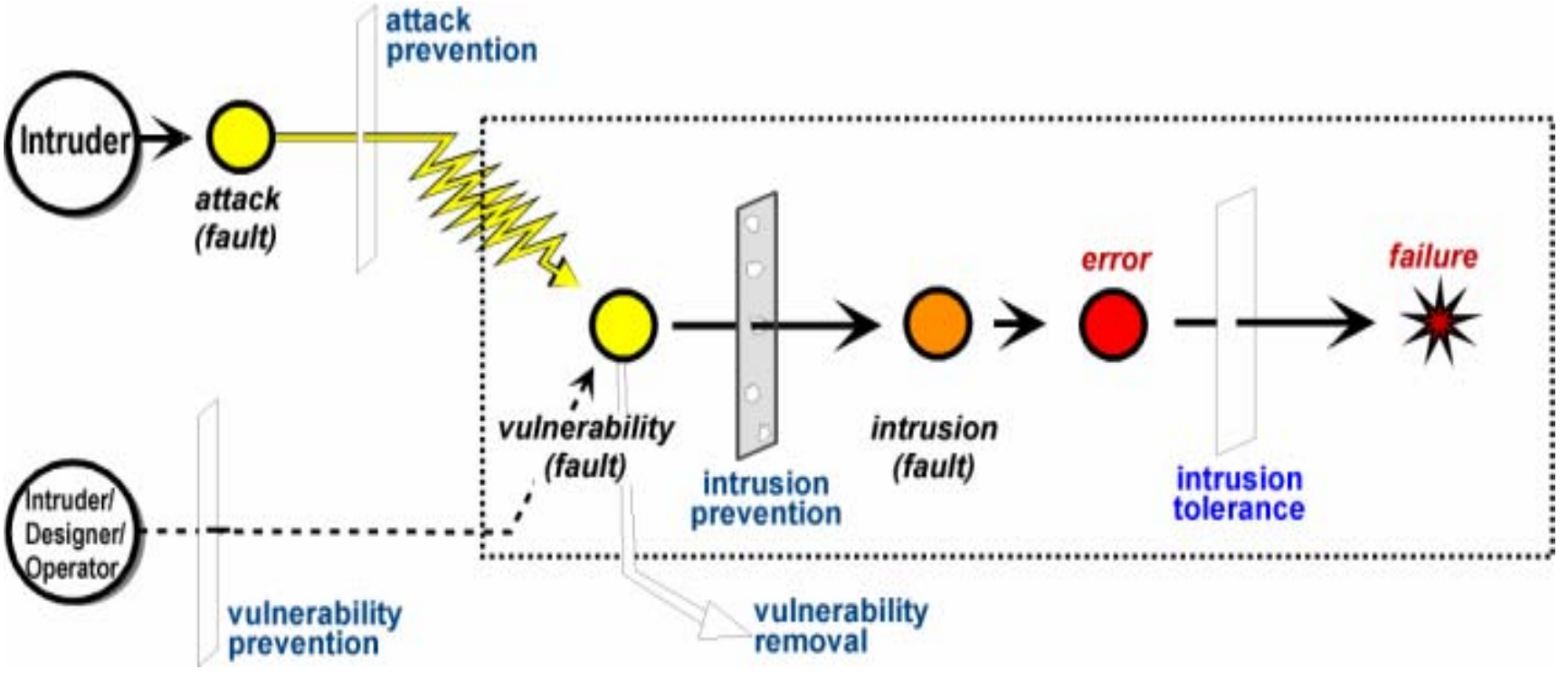
Maftia Workshop

February 2003, Newcastle U.

Grand Architectural Challenges

- **Failures**
  - what failure model?
- **Synchrony**
  - synchronous or asynchronous?
- **Topology**
  - what layers? trusted components?
- **Functionality**
  - what services?

Composite fault model



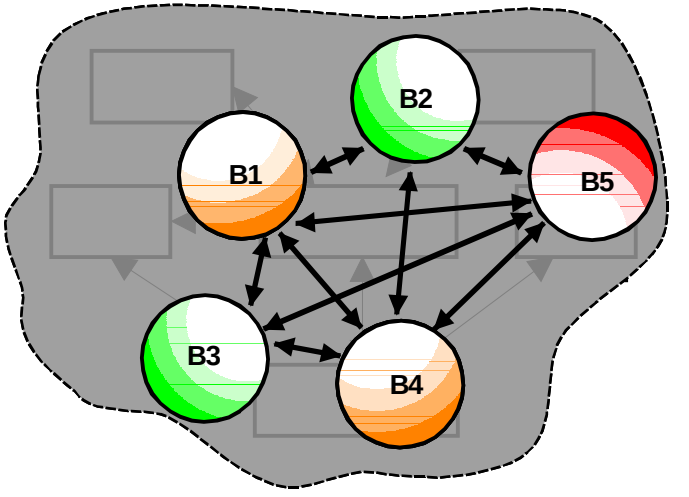
- sequence : *attack + vulnerability→ intrusion→ failure*

On Trust and Trustworthiness

- ***Thou shall not trust non-trustworthy components!***
  - A trusted component has a set of properties on which another component(s) depend...
- **Trust should be placed to the extent of that component's trustworthiness, the measure in which it meets those properties**
  - trust may have several degrees, quantitatively or qualitatively
  - related not only with security-relat. properties (e.g., timeliness)
  - trust and trustworthiness lead to complementary aspects of the design and verification process
- **when A trusts B, A assumes something about B. Trustworthiness of B measures the coverage of the assumption**

Building trust

- **Component-based approach**
- **Separation of concerns:**
  - higher level algorithms or assertions (e.g., authent/authoriz. logics);
  - infrastructure running them (e.g., procs/servers/comm's)
  - **Or**:
  - Component builder (trustworthiness)
  - Component user (trust)



Failure assumptions

- **Basic types of failure assumptions:**
  - **Controlled failures:**
  - assume qualitative and quantitative bounds on compon. failures, hard to substantiate for malicious faults
  - **Arbitrary failures:**
  - unbounded failures, limited only to the “possible” failures a component might exhibit (e.g. byzantine), playing on the safe side
  - **Hybrid failure assumptions**:
  - different assumptions for distinct component subsets, best of both worlds but how realistic?

Enforcing hybrid failure assumptions

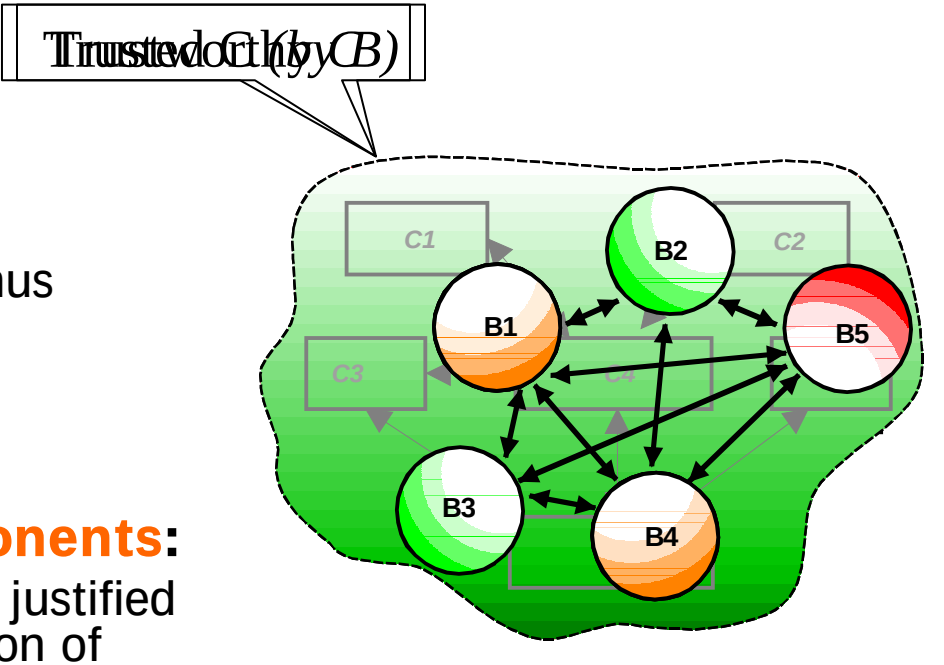
- **Component-based approach:**
  - modular architecture,
  - trust – trustworthiness relations between components
- **Composite fault model with hybrid failure assumptions:**
  - presence and severity of vulnerabilities, attacks, intrusions varies from component to compon.
- **How to achieve coverage, given unpredictability of attacks, and elusiveness of vulnerabilities?**

A robust design approach

- **Architectural hybridization:**
  - failure assumptions enforced by architecture and construction, thus substantiated
  - combined/recursive use of attack/vulnerability/intrusion prevention/removal/tolerance
- **Trusted (trustworthy) components:**
  - components or subsystems with justified coverage, used in the construction of fault-tolerant protocols under architectural hybrid failure assumptions

A robust design approach

- **Architectural hybridization:**
  - failure assumptions enforced by architecture and construction, thus substantiated
  - combined/recursive use of attack/vulnerability/intrusion prevention/removal/tolerance
- **Trusted (trustworthy) components:**
  - components or subsystems with justified coverage, used in the construction of fault-tolerant protocols under architectural hybrid failure assumptions



Enforcing hybrid failure assumptions

- **Component-based approach:**
  - modular architecture
  - trust – trustworthiness relations between components
- **Composite fault model with hybrid failure assumptions:**
  - the presence and severity of vulnerabilities, attacks and intrusions varies from component to component
  - how to achieve coverage of controlled failure assumptions, given unpredictability of attacks and elusiveness of vulnerabilities?

- **Design approach:**
- **Architectural hybridization:**
  - failure assumptions are in fact enforced by the architecture and the construction of the system components, and thus substantiated
  - Combined/recursive use of attack/vulnerability prevention/removal, intrusion prevention/tolerance
- **Trusted (trustworthy) components:**
  - components with justified coverage, used in the construction of fault-tolerant protocols under hybrid failure assumptions

Synchrony Model

- **Basic synchrony models:**
- **Synchronous models (timed):**
  - time, a powerful construct: solve timed problems
  - yield simple algorithms
  - susceptible to attacks on timing assumptions
- **Asynchronous model (time-free):**
  - resist attacks on timing assumptions
  - no deterministic solution of e.g. consensus, BA
  - efficient probabilistic approaches
  - do not solve timed problems (e.g., e-com, stocks)
- **Partial Synchrony**
  - exploit the power of intermediate models
  - accommodate several degrees of sync/async.

Where to go from here?

- **time versus resilience tradeoff:**
  - timed partially synchronous protocols *and* time-free protocols
- **efficiency versus resilience tradeoff:**
  - fail-controlled protocols *and* arbitrary failure protocols
- **trust vis-a-vis trustworthiness:**
  - high assumption coverage from component design
- **incremental fault tolerance:**
  - mech's and prot's providing range of resilience degrees

Intrusion-tolerance design strategies in MAFTIA

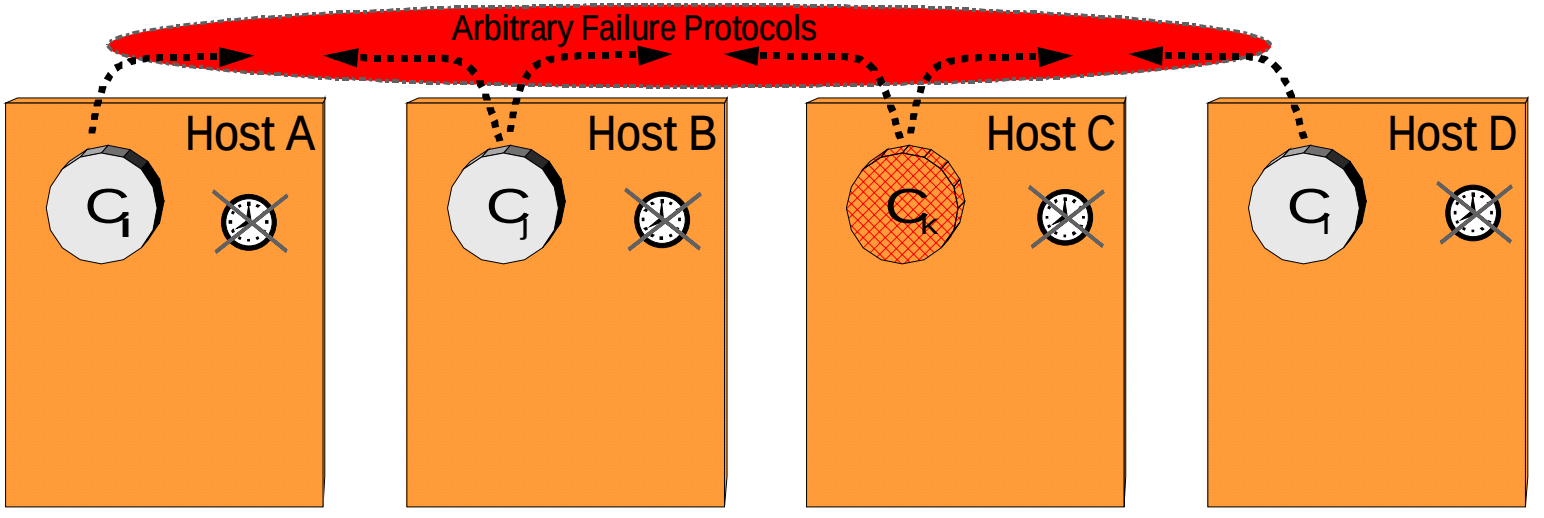
- **Fail-uncontrolled**
- **Fail-controlled with trusted components**
  - local
  - distributed

Arbitrary failure assumptions considered necessary

- **operations of high value and/or criticality:**
  - risk of failure due to violation of assumptions cannot be incurred
- **arbitrary-failure resilient building blocks (e.g. Byzantine agreement protocols):**
  - no assumptions on existence of security kernels or other fail-controlled components, or about timeliness

Fail-uncontrolled

- **Time-free**
- **Arbitrary failure environment**
- **Arbitrary failure protocols**
- **Used in: probabilistic Byzantine-agreement based set of protocols**

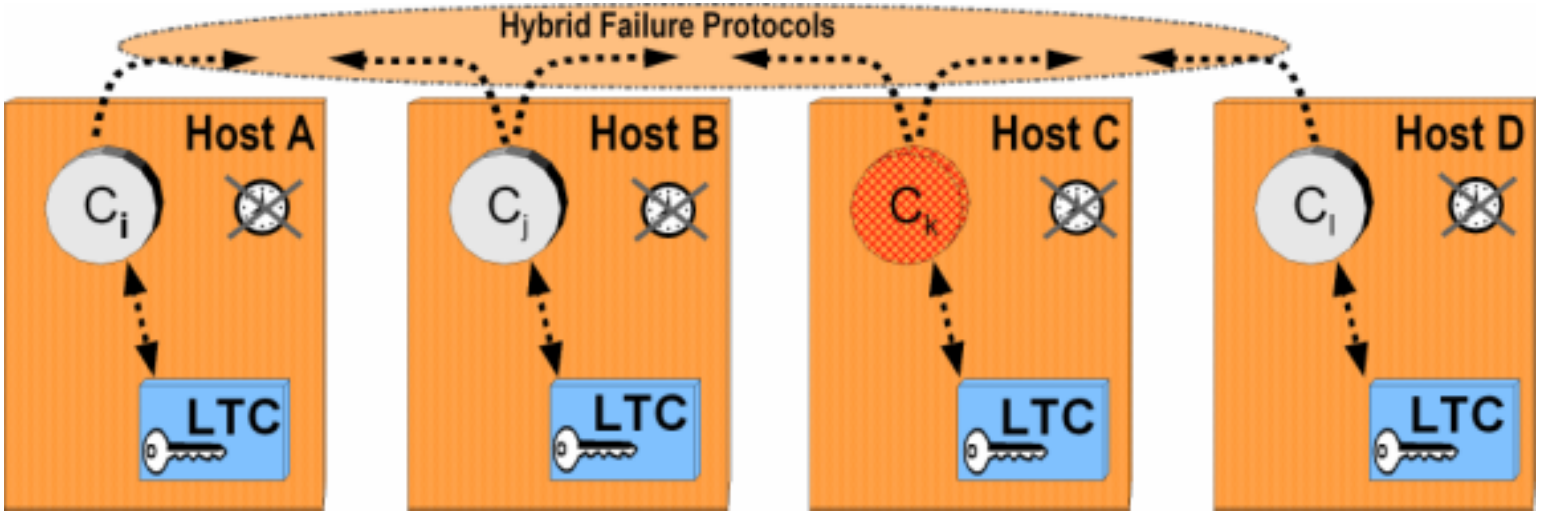


Intrusion tolerance with hybrid failure assumptions

- **Implementing small trustworthy components…**
  - trusted to different extents
  - trusted to execute simple but crucial functions correctly
- **…used in fault-tolerant protocols:**
  - more efficient than truly arbitrary assumptions protocols
  - more robust non-enforced controlled failure protocols
- **Two instances of such trusted components:**
  - *local TC*, based on a Java Card module, designed to assist the crucial steps of the execution of services and applications.
  - *distributed TC* (Trusted Timely Computing Base), based on appliance boards with private network adapters, designed to assist crucial steps of middleware protocols.

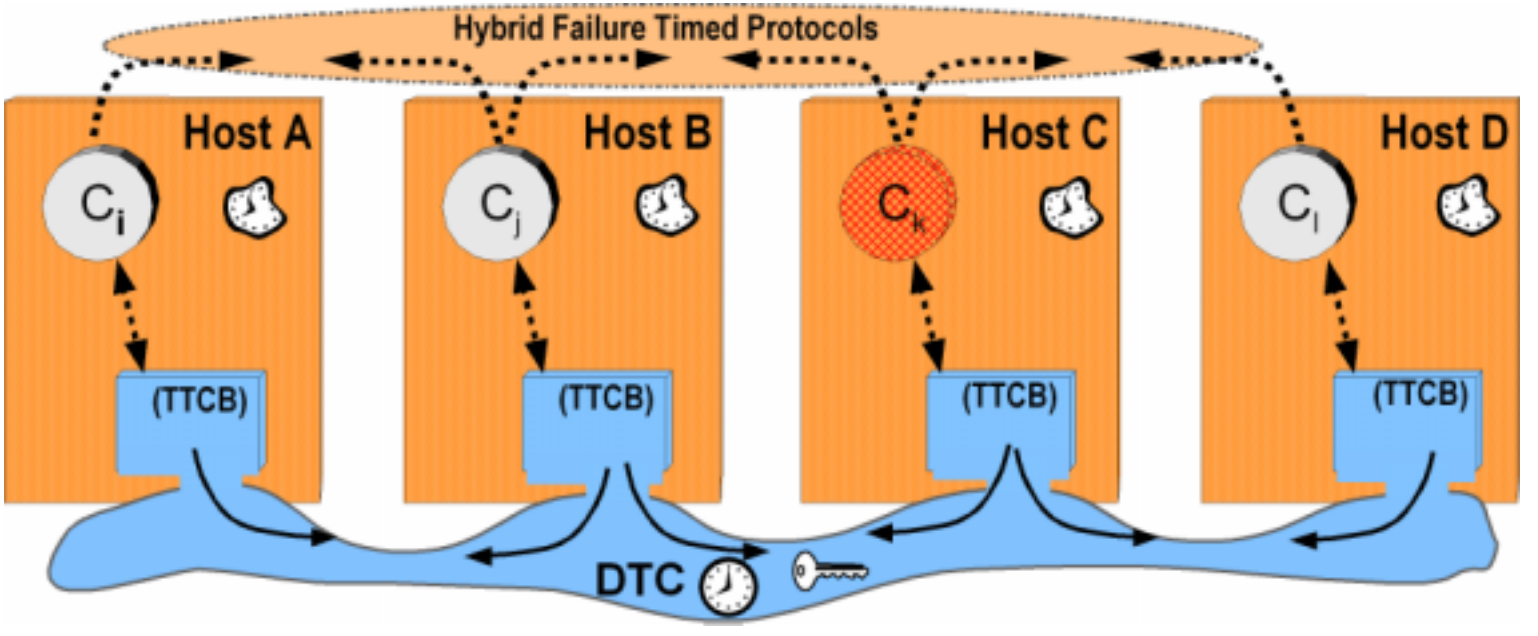
Fail-controlled with Local trusted components

- **Time-free**
- **Arbitrary failure environment + LTC**
- **Hybrid failure protocols**
- **Used in: construction of the authorisation service**
- **Trusted to the extent of: presenting certain hardness to being broken, and of operating correctly until then**



Fail-controlled with Distributed trusted components

- **Time-free or timed with uncertain synchrony**
- **Arbitrary failure environment + synchronous DTC**
- **Hybrid failure protocols**
- **Used in: construction of malicious-F-T comm's protocols**
- **Trusted to the extent of: not being feasible to subvert it**



Time-free Programming model

- **Stand-alone programming model:**
  - fully-asynchronous (time-free) setting
  - arbitrary failures
  - randomized probabilistic solutions
- **TTCB-supported programming model:**
  - payload system is time-free and suffers malicious faults
  - time-free fully asynchronous payload appl/protocols
  - TTCB performs time-based error detection and supplies basic security functions

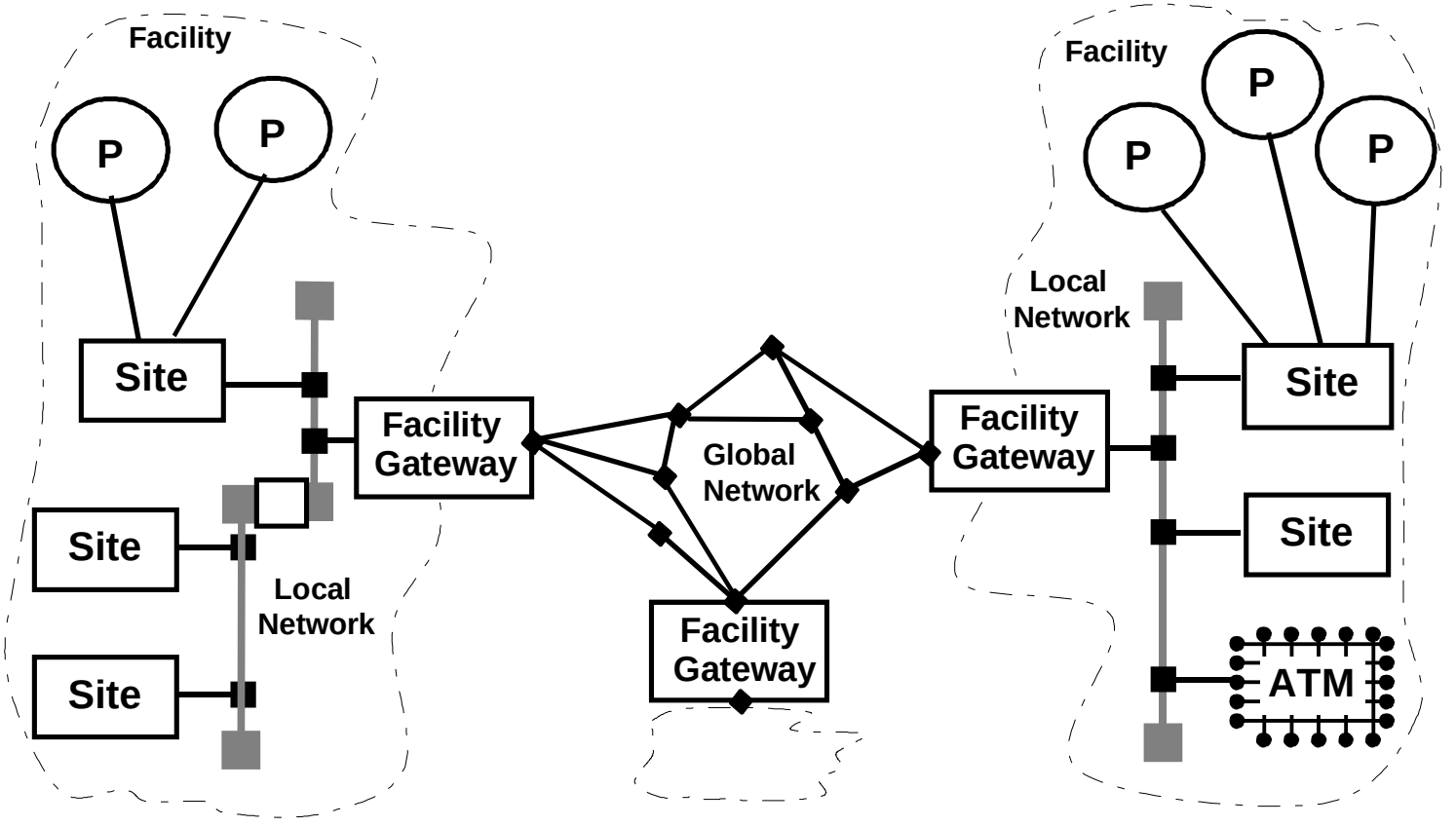
Timed Programming model

- **The payload system is timed**
  - but has uncertain timeliness and suffers malicious faults
- **The control system (the TTCB), can assist an application running on the payload system**
  - determine useful facts about time (be sure it executed something on time; measure a duration; determine it was late doing something), and supply basic security functions
- **The payload system, despite imperfect can act/react (fault tolerance mechanisms)**
  - based on reliable information about presence or absence of errors, provided by the TTCB at its interface

Architecture Overview

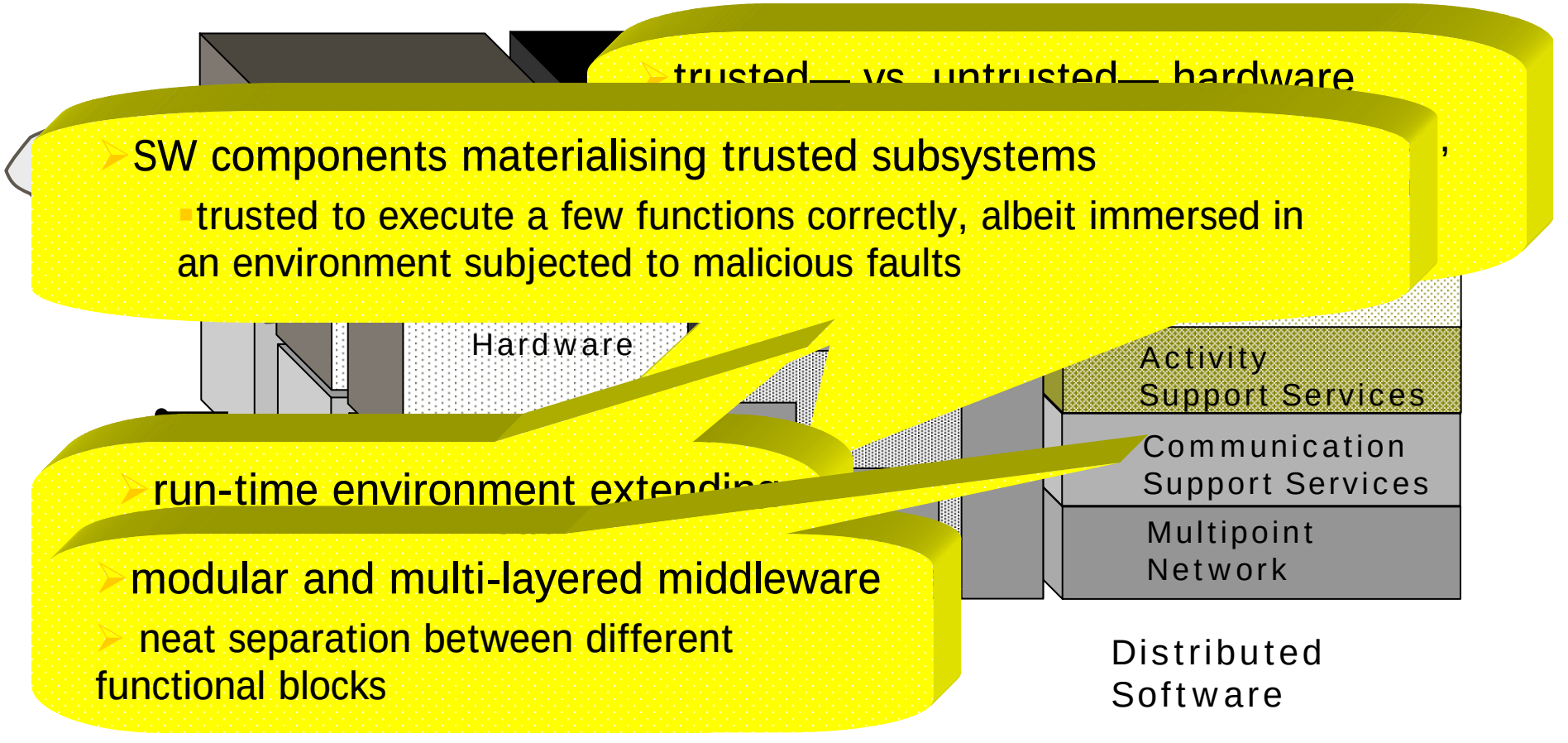
Architecture Overview

View from afar: WAN-of-LANs structure



Architecture Overview

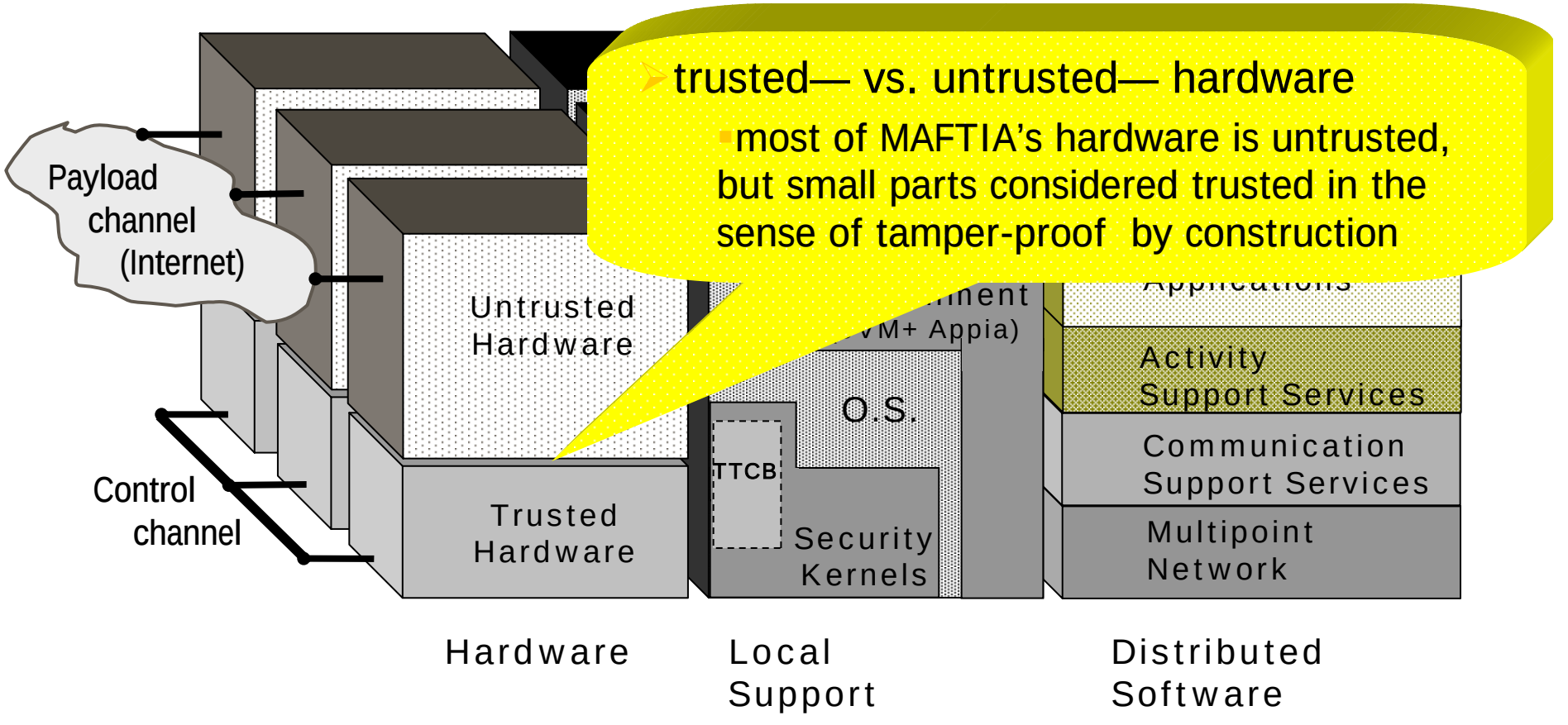
Host architecture



AS - Authorisation Service, IDS - Intrusion Detection Service, TTP - Trusted Third Party Service

Architecture Overview

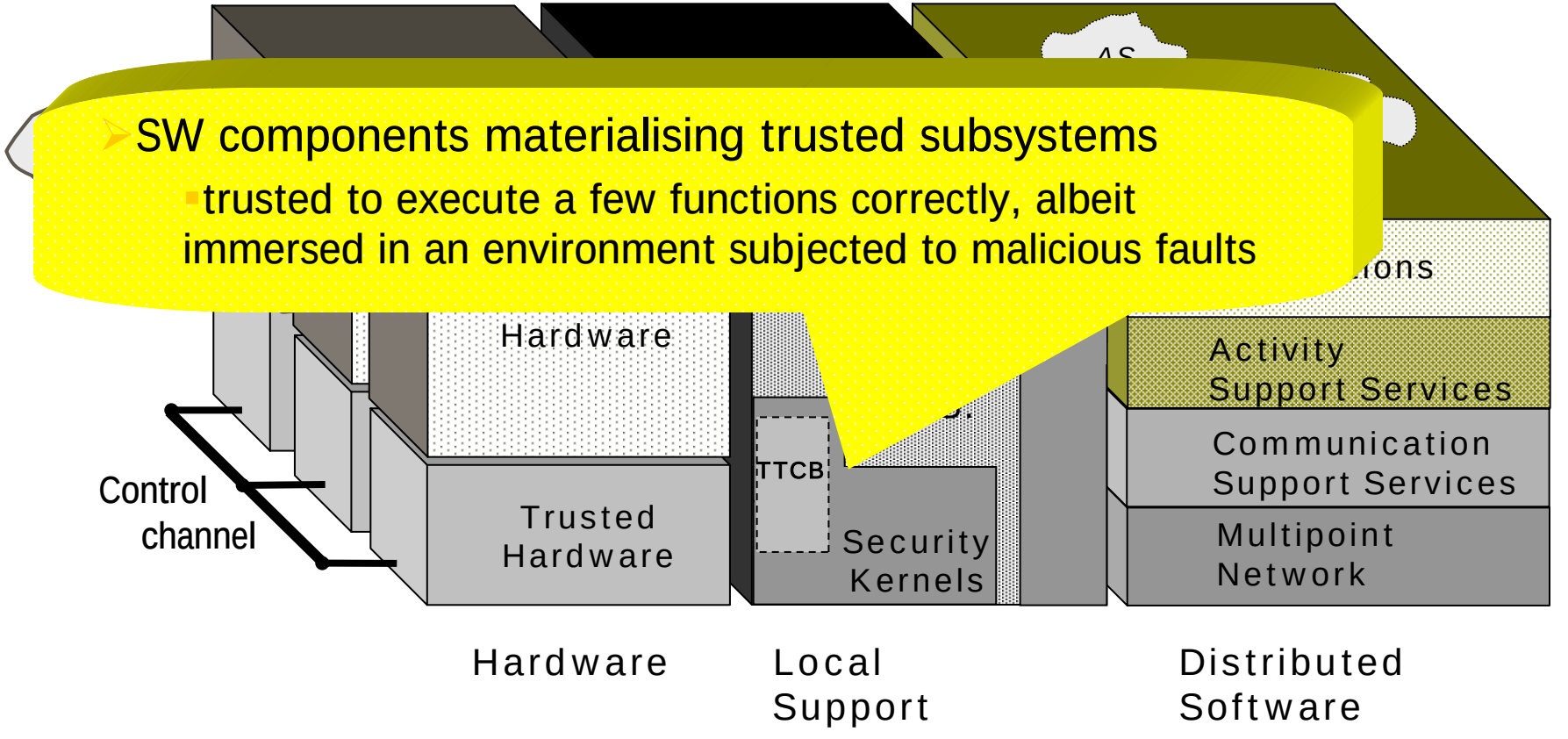
Host architecture



AS - Authorisation Service, IDS - Intrusion Detection Service, TTP - Trusted Third Party Service

Architecture Overview

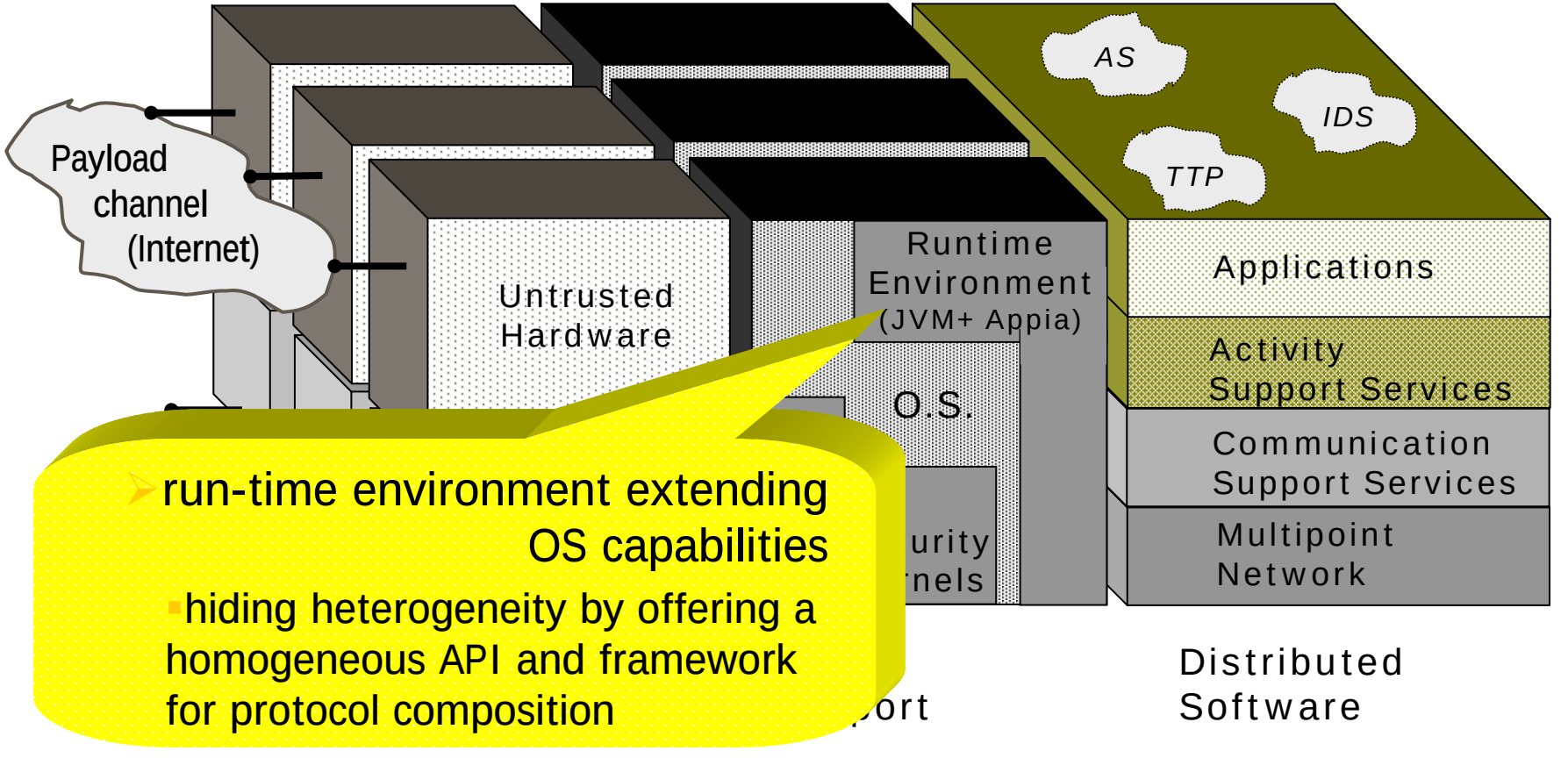
Host architecture



AS - Authorisation Service, IDS - Intrusion Detection Service, TTP - Trusted Third Party Service

Architecture Overview

Host architecture

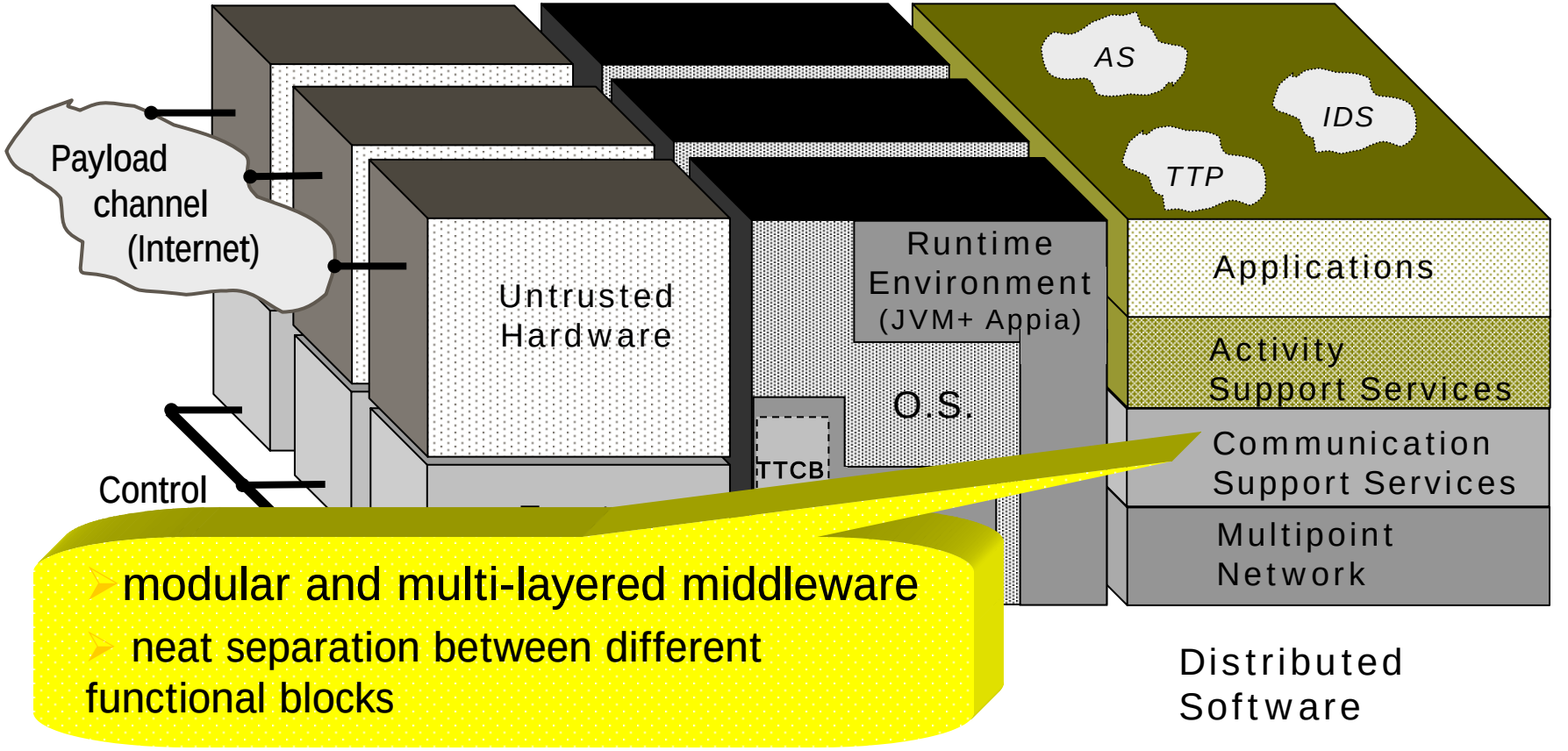


- run-time environment extending OS capabilities
  - hiding heterogeneity by offering a homogeneous API and framework for protocol composition

AS - Authorisation Service, IDS - Intrusion Detection Service, TTP - Trusted Third Party Service

Architecture Overview

Host architecture



AS - Authorisation Service, IDS - Intrusion Detection Service, TTP - Trusted Third Party Service

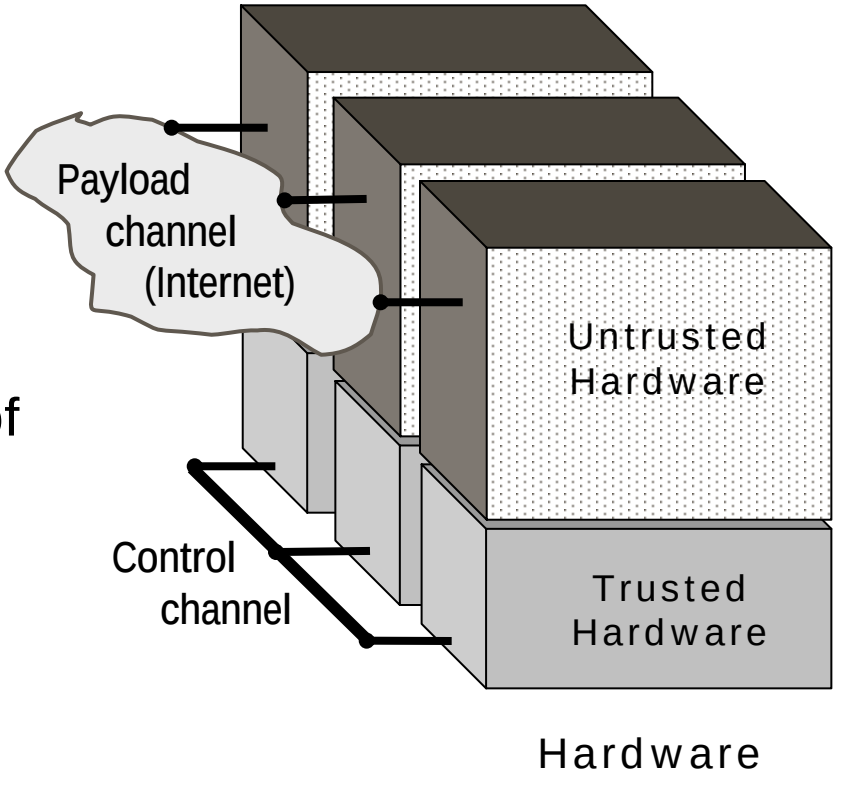
Architecture Overview

Main architectural options

- **trusted — versus untrusted — hardware**
  - most of MAFTIA's hardware is untrusted, but small parts considered trusted in the sense of tamper-proof by construction
- **trusted SW in the run-time support**
  - trusted to execute a few functions correctly, albeit immersed in an environment subjected to malicious faults
- **run-time environment extending OS capabilities**
  - hiding heterogeneity by offering a homogeneous API and framework for protocol composition.
- **modular and multi-layered middleware**
  - neat separation between different functional blocks

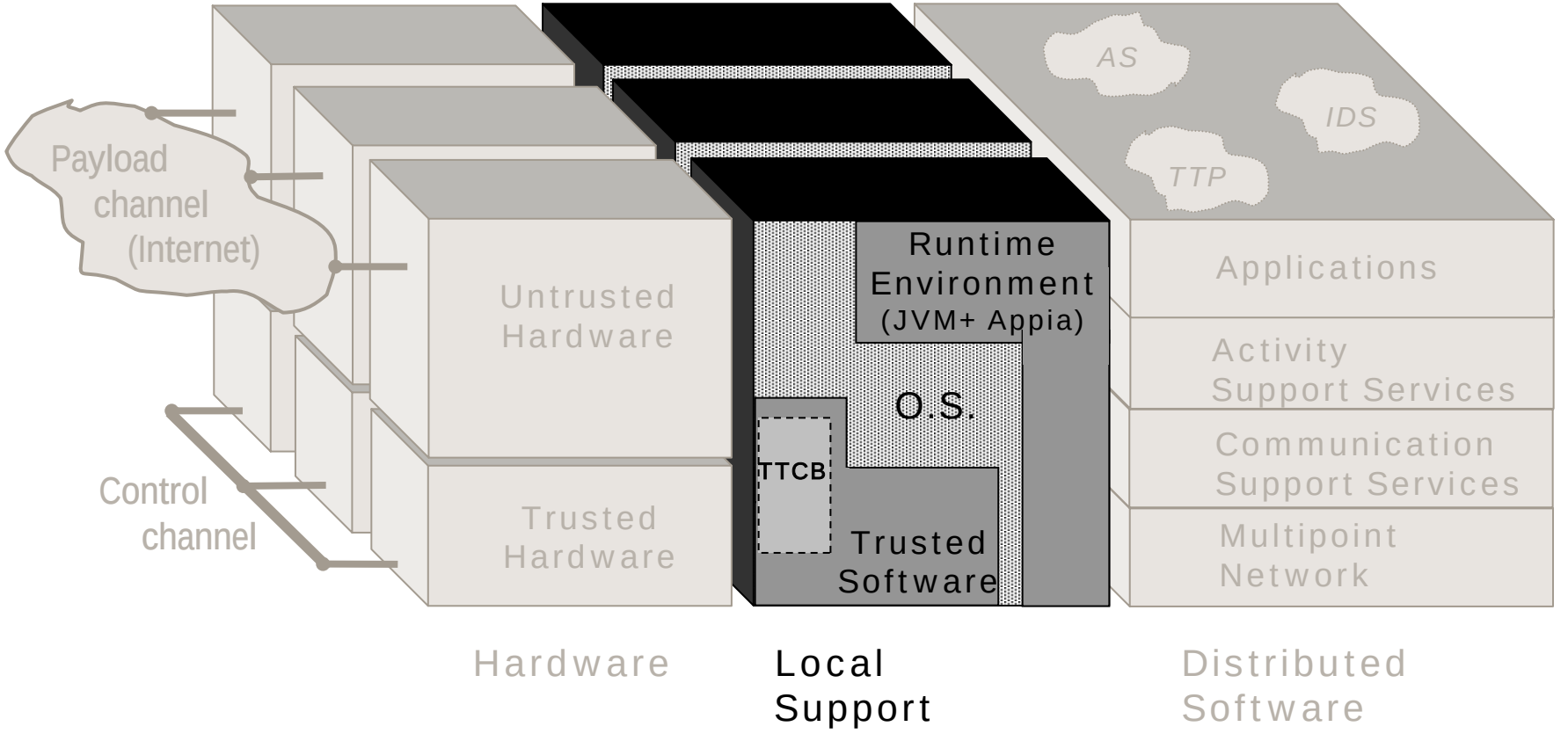
Hardware

- **untrusted hardware:**
  - usual machinery of a PC or workstation, normal networking infrastructure (*payload channel)*
- **trusted hardware:**
  - tamper-proof--- intruders do not have direct access to the inside of the component
  - Made of COTS components:
  - ***Java Card reader***
  - ***Appliance board***
  - ***Control channel***



Architecture Overview

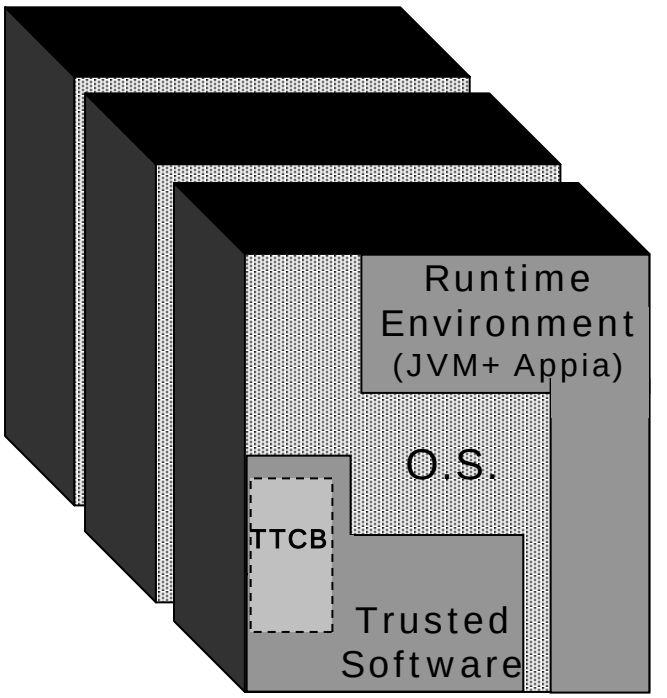
Host architecture



AS - Authorisation Service, IDS - Intrusion Detection Service, TTP - Trusted Third Party Service

Local run-time support

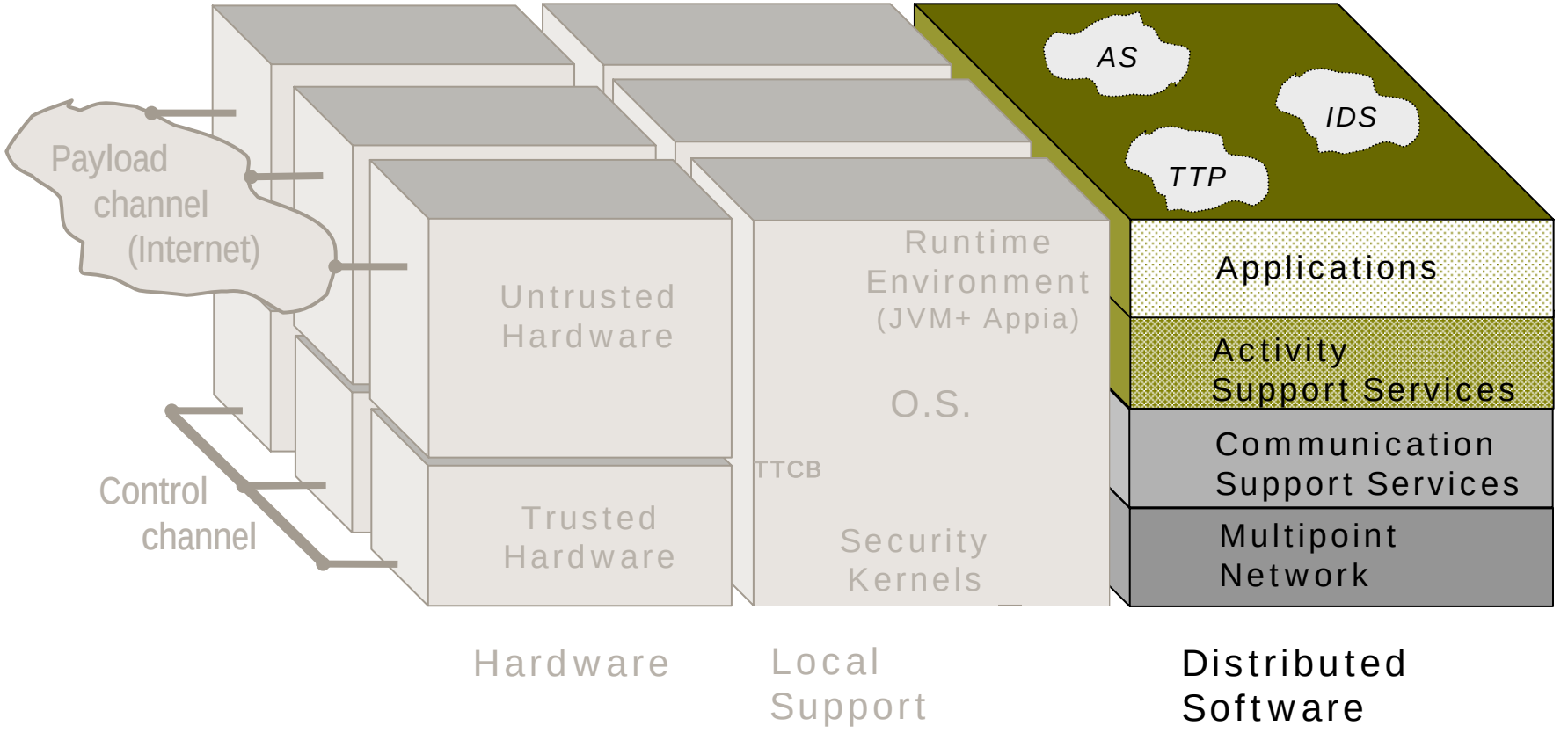
- **OS augmented with extensions**
  - Java Virtual Machine (JVM)
  - APPIA protocol composition kernel
  - special functions from security kernels
- **Java Card module**
  - checks accesses to local objects
  - manages all access rights for local transient objects
- **Trusted timely computing base (TTCB)**
  - basic set of low-level trusted services related to time and security
  - supports intrusion-tolerance and timeliness
  - acts as an oracle that participants can trust



Local
Support

Architecture Overview

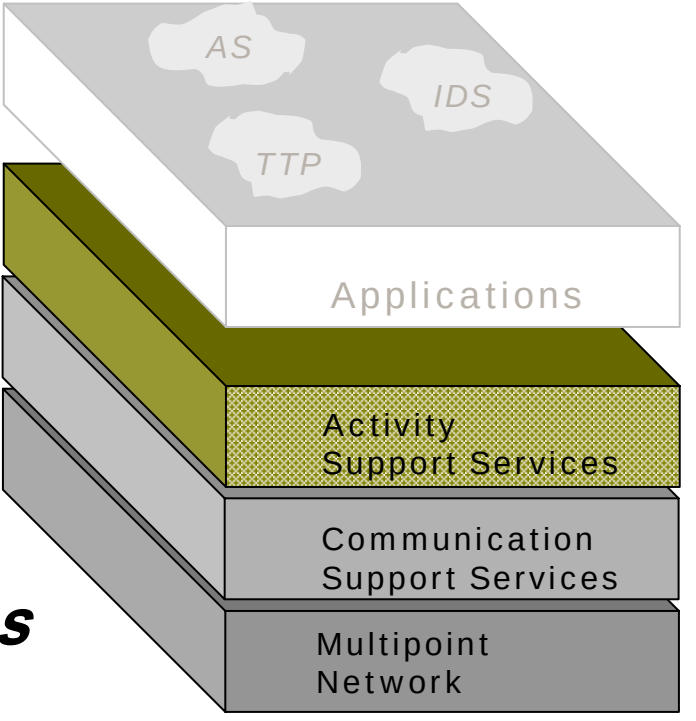
Host architecture



AS - Authorisation Service, IDS - Intrusion Detection Service, TTP - Trusted Third Party Service

Middleware

- **composition of micro-protocols**
- **uniform APIs**
- **different pgm'ing profiles**

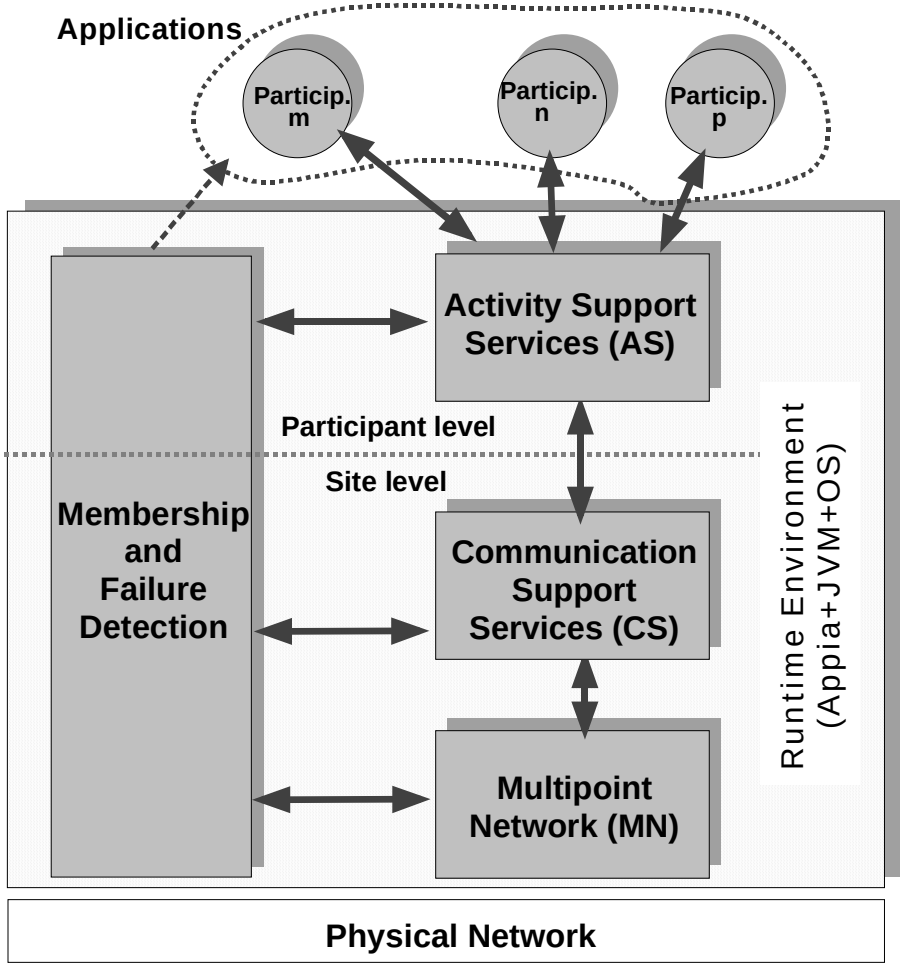


Distributed Software

- ***Activity Support Services***
  - assist participant activity
- ***Communication Support Services***
  - implement secure group comm's
- ***Multipoint Network***
  - adapts physical infrastructure

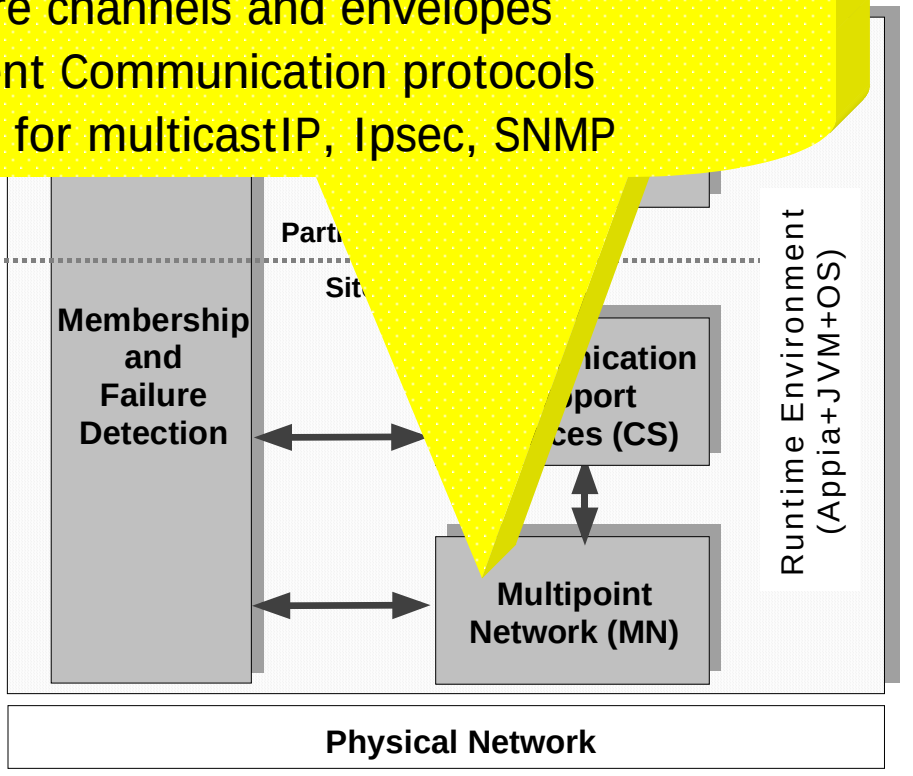
Modular Group-oriented Architecture

- **Site part:**
  - Takes care of host-to-host communication
- **Participant part:**
  - Takes care of distributed activity of processes



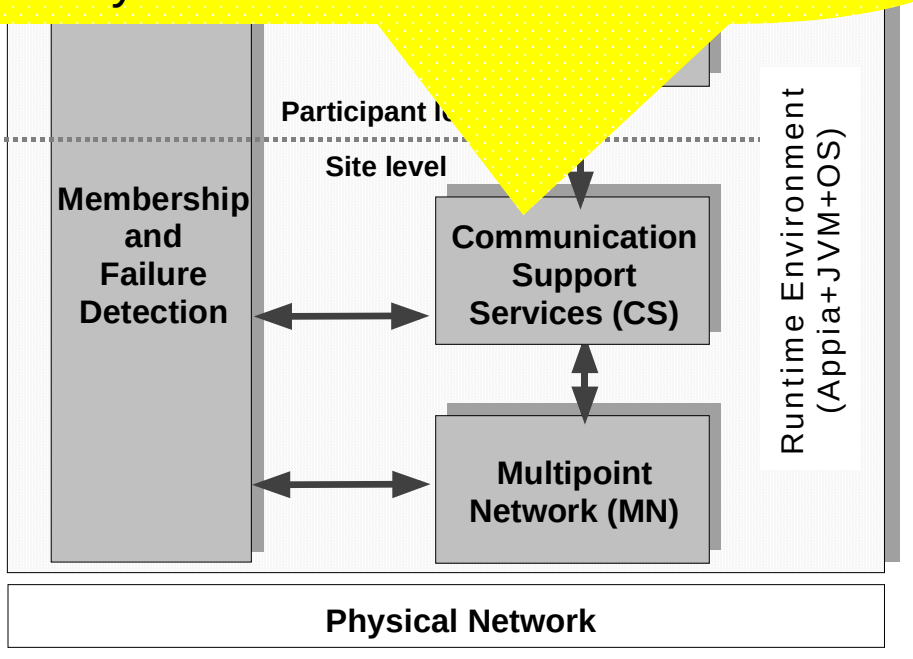
Modular Group Architecture

- Multipoint Network
  - Multipoint addressing and routing
  - Basic secure channels and envelopes
  - Management Communication protocols
  - Appia APIs for multicastIP, Ipsec, SNMP

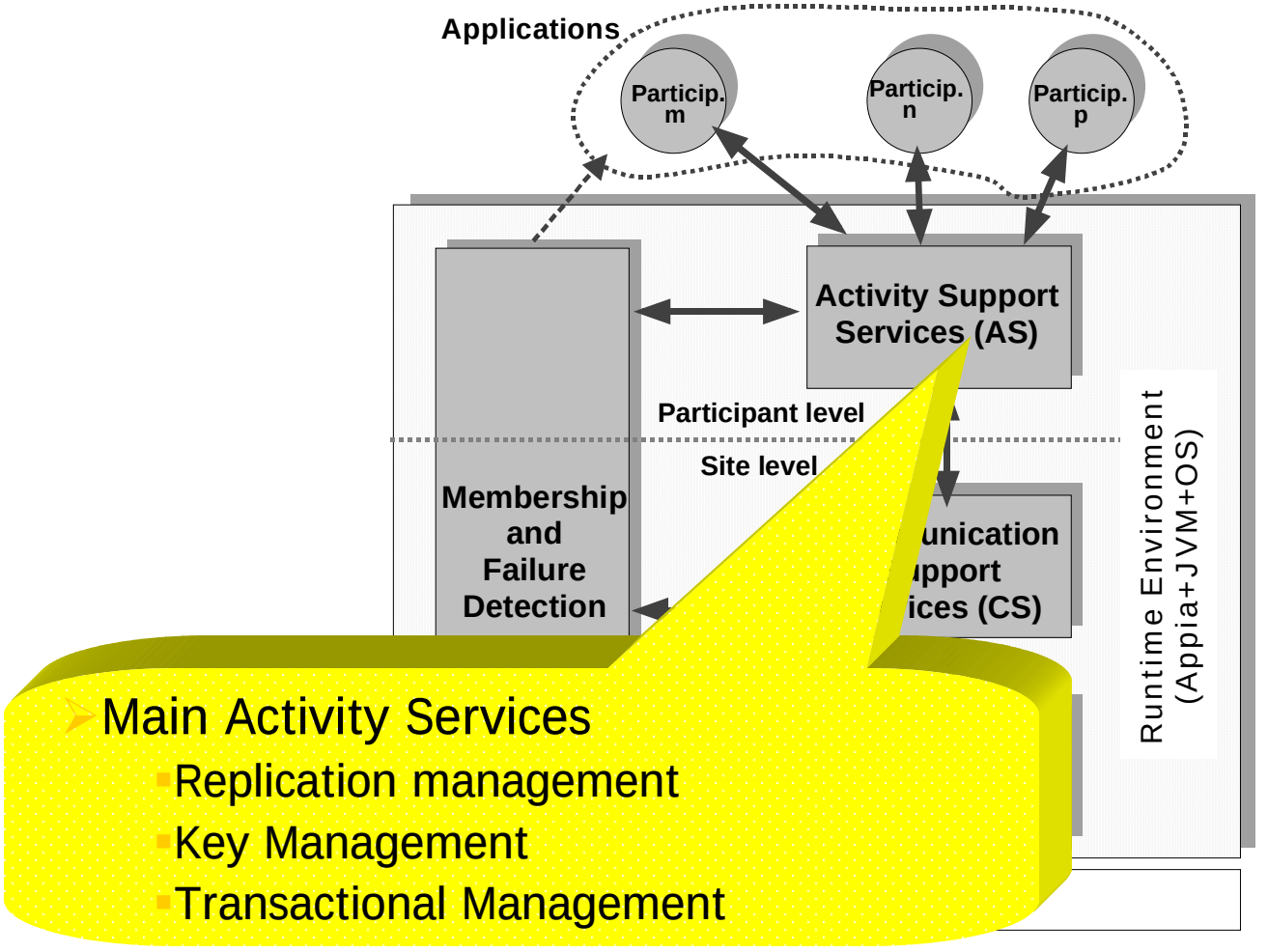


Modular Group Architecture

- Communication Services
  - Distributed Cryptography (threshold public key prots)
  - Group Communication (reliability and ordering props)
  - Byzantine Agreement
  - Time and Clock Synchronisation



Modular Group Architecture

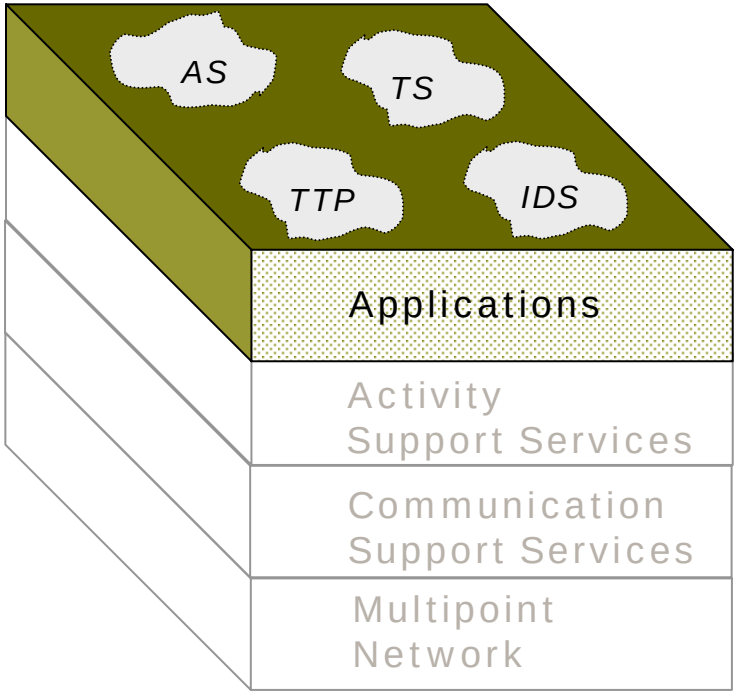


Middleware Services

- **Multipoint Network**
  - Multipoint addressing and routing
  - Basic secure channels and envelopes
  - Management Communication protocols
  - Essentially Appia APIs for multicastIP, Ipsec, SNMP
- **Communication Services**
  - Distributed Cryptography (threshold public key protocols)
  - Group Communication (several reliability and ordering props)
  - Byzantine Agreement
  - Time and Clock Synchronisation
- **Main Activity Services**
  - Replication management
  - Key Management
  - Transactional Management

Examples of MAFTIA intrusion-tolerant services

- **Authorisation Service**
- **Intrusion Detection Service**
- **Transaction Service**
- **Trusted-Third-Party Service**



IT Services

The End