A new Distributed Security Model for Linux Clusters

Makan.Pourzandi@Ericsson.Com

Open Systems Lab

Montréal – Canada

June, 2004

Outline

- Context
- Distributed Security
- Distributed Access Control (DisAC)
- DisAC implementation: Distributed Security Infrastructure

Context

- Target application:
  - Large distributed applications
  - Large software base
  - Up 24 hours/7 days
  - High Availability: from 2 to 5 nines (99.999%) uptime
- Linux Clustered servers
- Providing services to different operators
- Exposed to the Public network/Internet
- Running untrusted third-party software
  - No access to the sourc code
  - No possibility to audit the code for economical and schedule related reasons
- Software configuration evolves slowly over time: no wild software installations

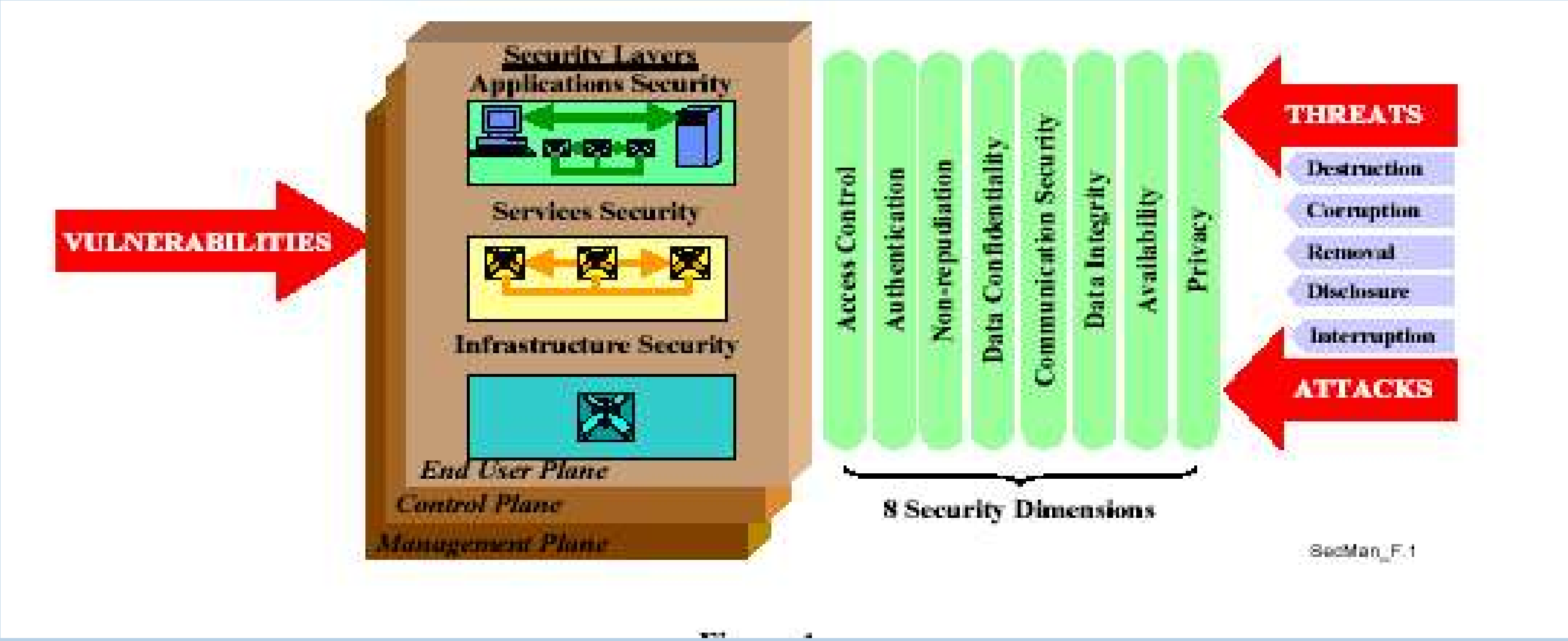
Specific Security Needs

- Security isolation, compartmentalization
  - Large applications supporting many functionalities
  - Probability of exploitable vulnerabilities in a large software base
  - A vulnerability in some parts of the system can compromise the entire system
- Pre-emptive security: changes in the security context will be reflected immediately
- Dynamic security policy: run time changes in security context and policy
- Don't only rely on application layer security mechanisms: “Administrators must contend with vulnerabilities in applications over which they have no direct control”

Distributed Security

Challenges in Distributed security

- Implement coherent distributed security
  - Many layers to **fit together**: Applications, Middleware, OS, Hardware, Network …
  - Heterogeneous environment: variety of Hardware, Software: OS, Middleware, Netwozrking technologis



Security Architectural Elements in ITU-T X.805,
from Security in Telecom and Information Technology, Dec 2003

Challenges in Distributed security (2)

- Integration of different security solutions from potentially different vendors...
- System management
  - If manually managed, it may lead to misconfigurations and inconsistencies

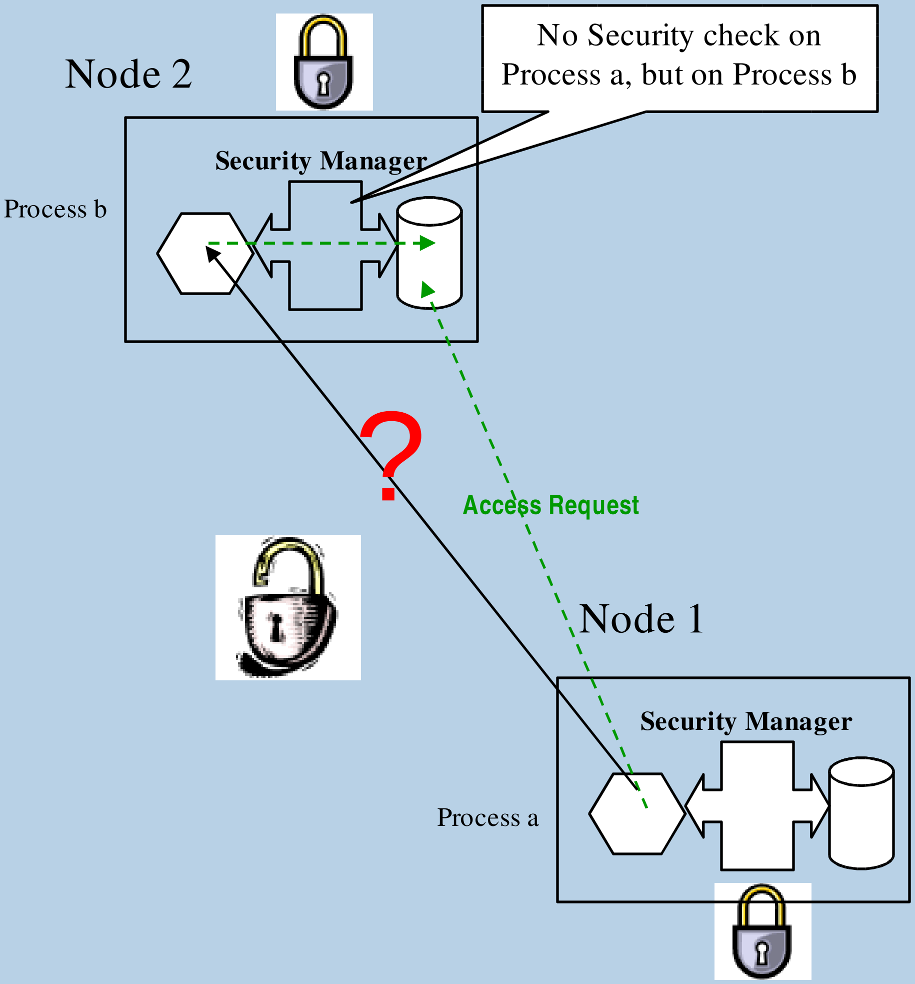
“Distributed Systems Require **Distributed Security**”

Hartman, Flinn, Beznosov,

Enterprise Security with EJB and CORBA

User based access control approach on cluster computing

- Based on user privileges (login, password)
  - Life time: a session of several hours/days
  - Scope: limited range of operations according to the application's nature
  - Security policy based upon login and passwords
- Our target application:
  - Few users only
  - Life time: months if not years
  - Scope: wide range of operations, from upgrading software to managing information in database



Needs vs. Existing mechanisms

Current mechanisms

- Security policy based upon login and passwords
- Compartmentalization: necessity of creating/managing many users and groups
- User authentication often only at login time
- Running for short period of time (days) before each reboot
- No pre-emptive security

Needs

- Security policy depends on the processes and the source of the request rather than the user
- Running for a very long time (months) under the same login without rebooting
- Fine grained security policy based on processes
- Pre-emptive security

MAC vs DAC

- Discretionary Access Control (DAC)
  - It is at the discretion of the user to define access control privileges for user resources
- Mandatory Access Control (MAC)
  - Access control no longer solely depends on the user's decision
  - The system administrator defines the access control policy of the system

Existing solutions

- Many existing security solutions exist:
  - As external security mechanisms to the servers such as firewalls and Intrusion Detection Systems
  - As part of servers such as Integrity checks and some mechanisms to enhance security as a part of OS…
- Recent efforts
  - Linux Security Modules
  - SE Linux
  - Process right management in Solaris 10
- However, there are few efforts to make a coherent framework for enhancing security in a distributed system

Distributed Access Control (DisAC)

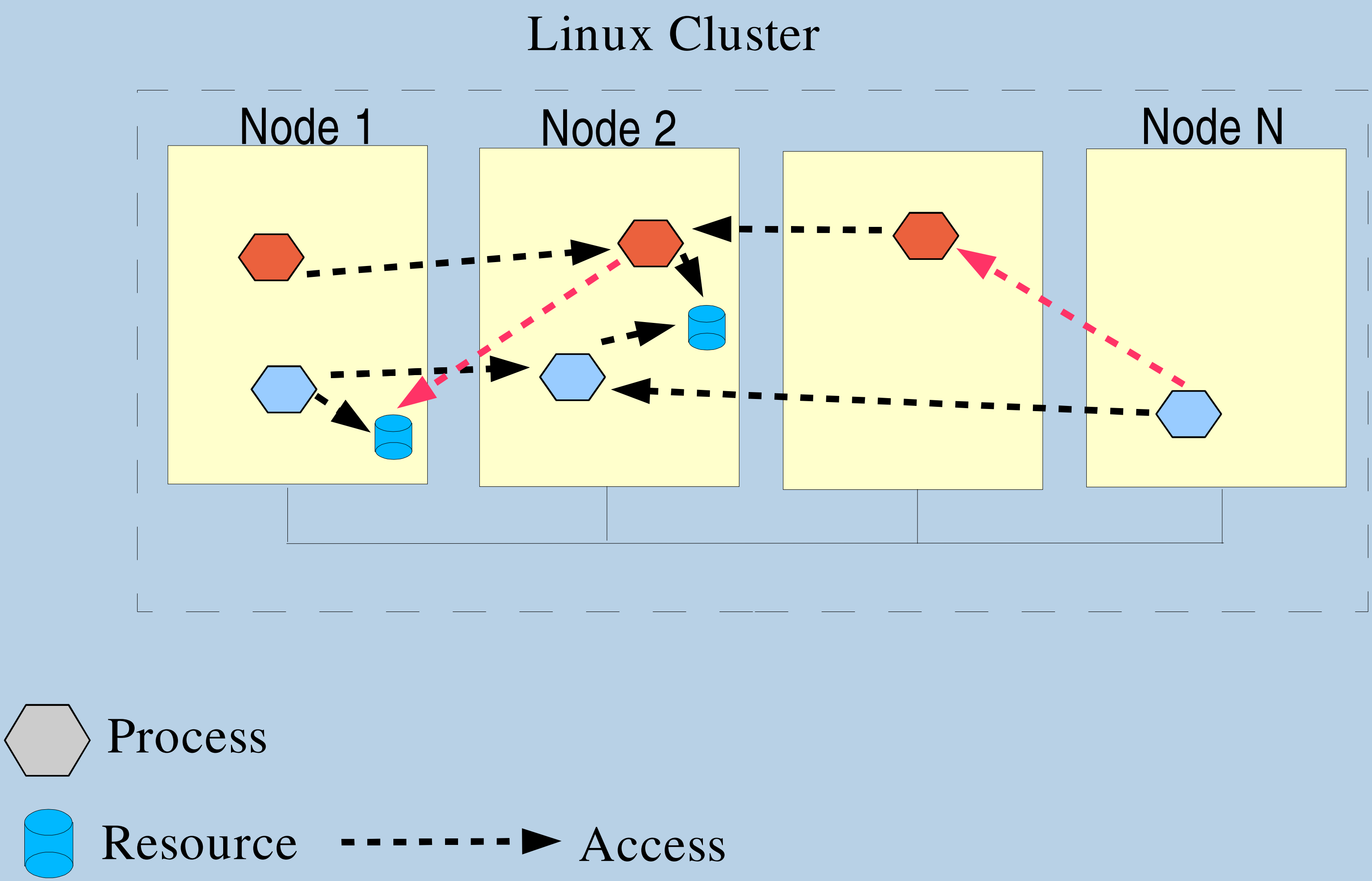
DisAC

- Goal
  - Extending kernel-level Mandatory Access Control features for a single computer into features for a distributed system
- Usage
  - Sharing the same cluster among different applications running untrusted third-party software
  - Compartmentalization: Setting up virtual security zones inside the cluster

Characteristics

- Cluster-wide access control
- Process-level granularity access control
- Access control at Operating system kernel level (Linux)
- Extra security functionality in addition to
  - User level authentication
  - Discretionary Access Control

General access model



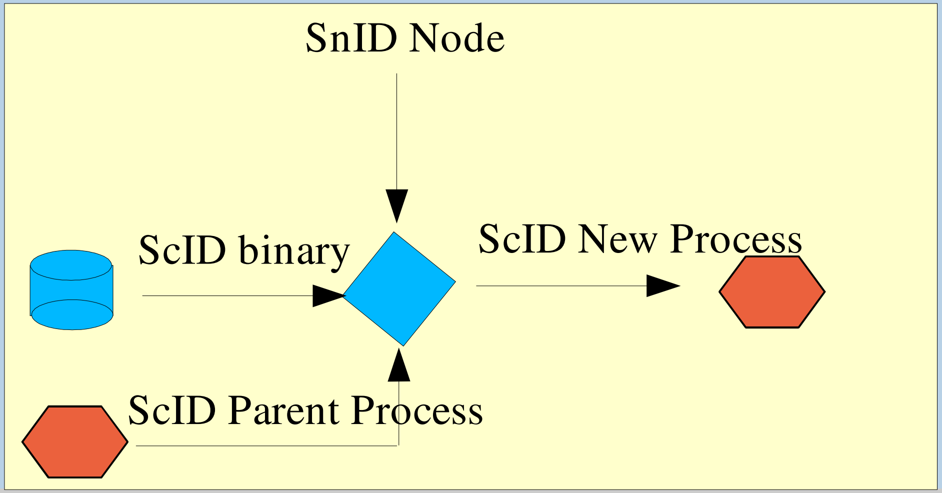
Process level Security Context

- Process security ID: ScID
  - Privileges associated with each process
    - Defined for the entire cluster: it can be transferred and interpreted through the whole cluster
    - Persistent
  - Security ID: corresponding to the group security context
    - Assigned by local security manager at the creaton of the process
- Node security ID: SnID
  - Privileges associated with each node
  - Unique for each node in the cluster
- Security Context for each process
  - Defined by: <ScID, SnID>

ERICSSON

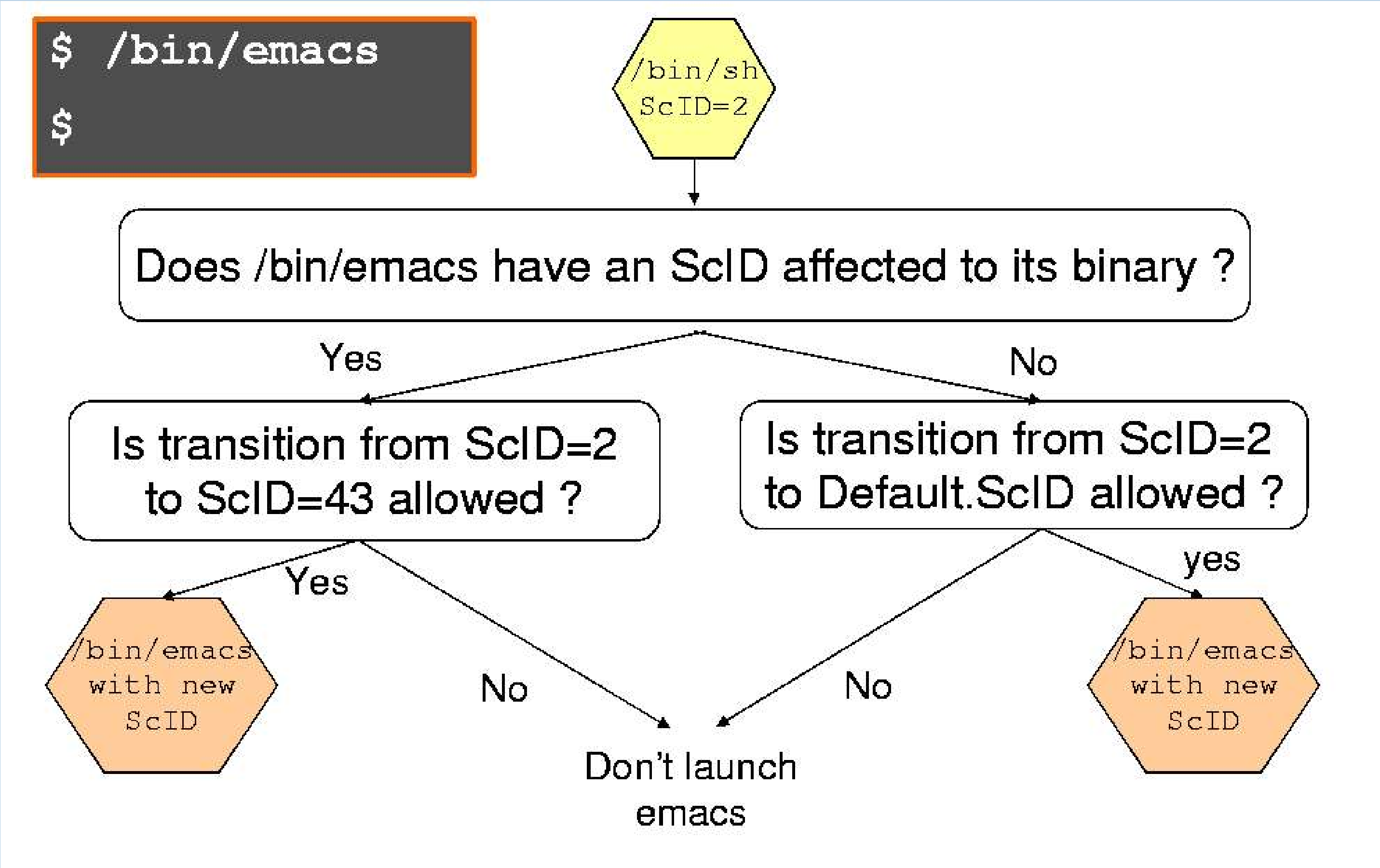
ScID for a newly created process

- ScID of binary:
  - Stored in ELF header
  - Integrity protected by digital signatures
- Parent Process ScID
- Node SnID
- New ScID stored at kernel level for each process



Classifying binaries based on security

- Using ScIDs for categorizing binaries



Access control decisions

- Process level based on <SnID, ScID> of source and target

- Locality in the distributed system is taken into account

- An access is only granted if the following access policy exists:
  - Source <SnID, ScID>, Target <SnID, ScID>, Allowed Action

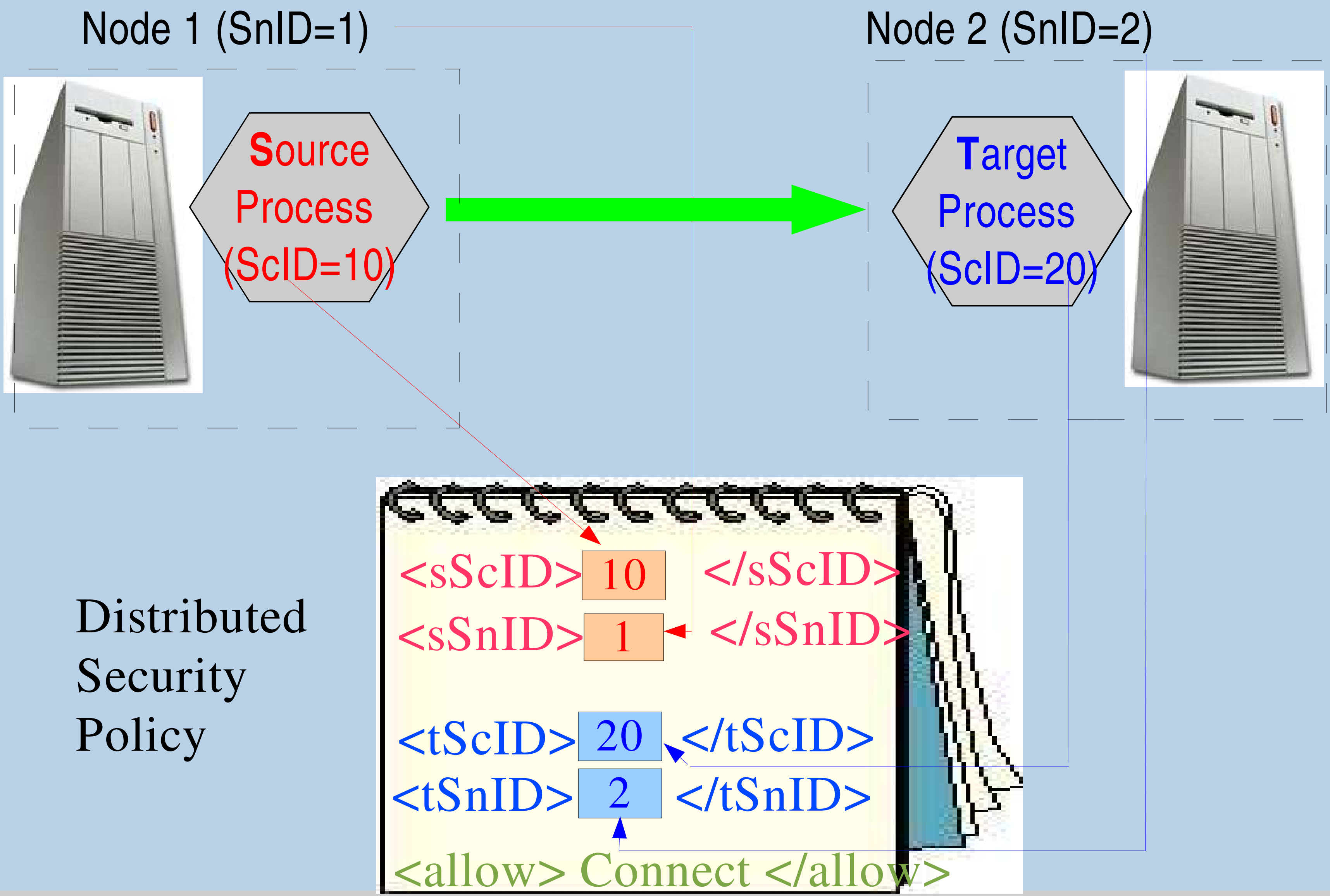
- Different classes of rules: network, socket, process, transition

ERICSSON

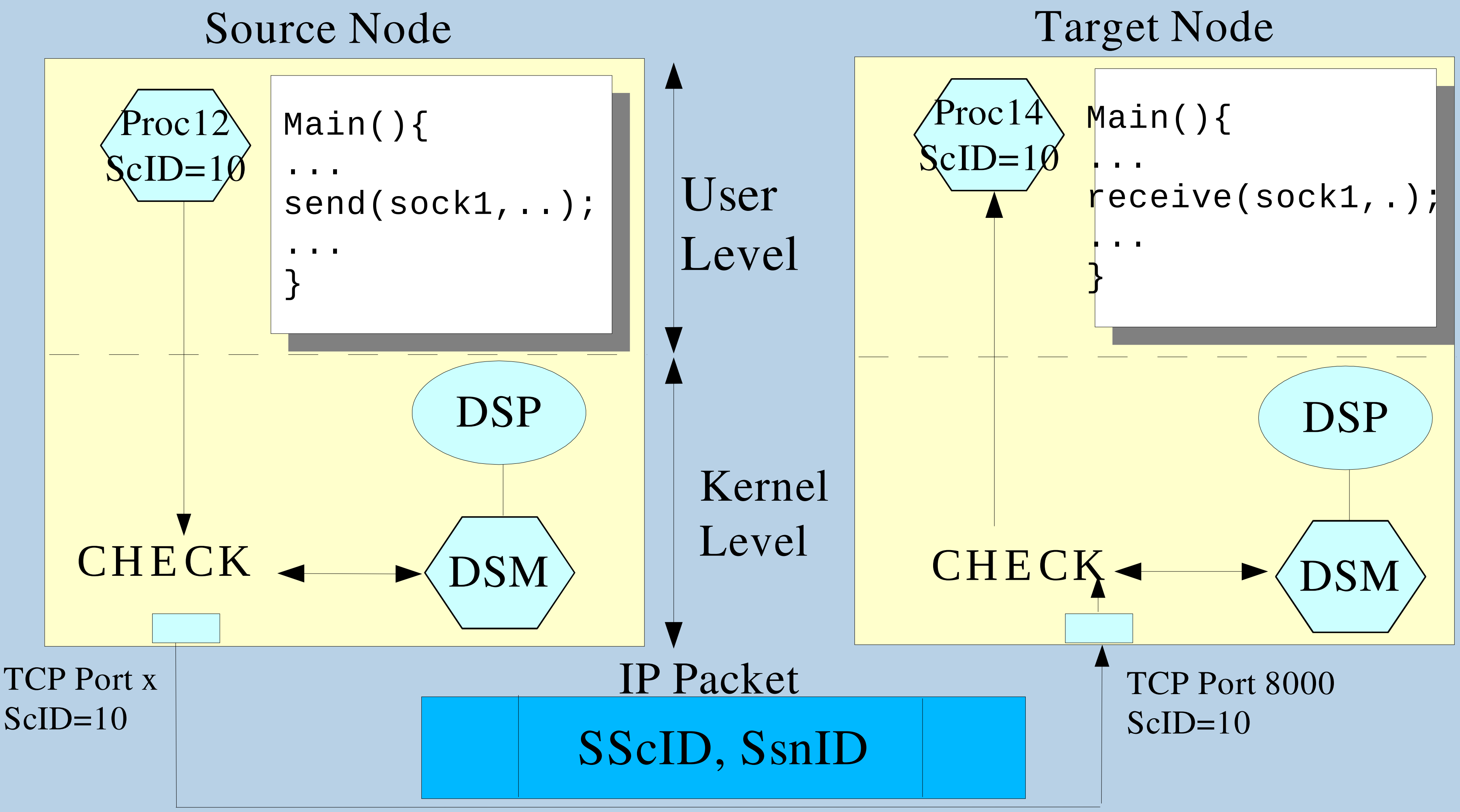
Distributed Security Policy (DSP)

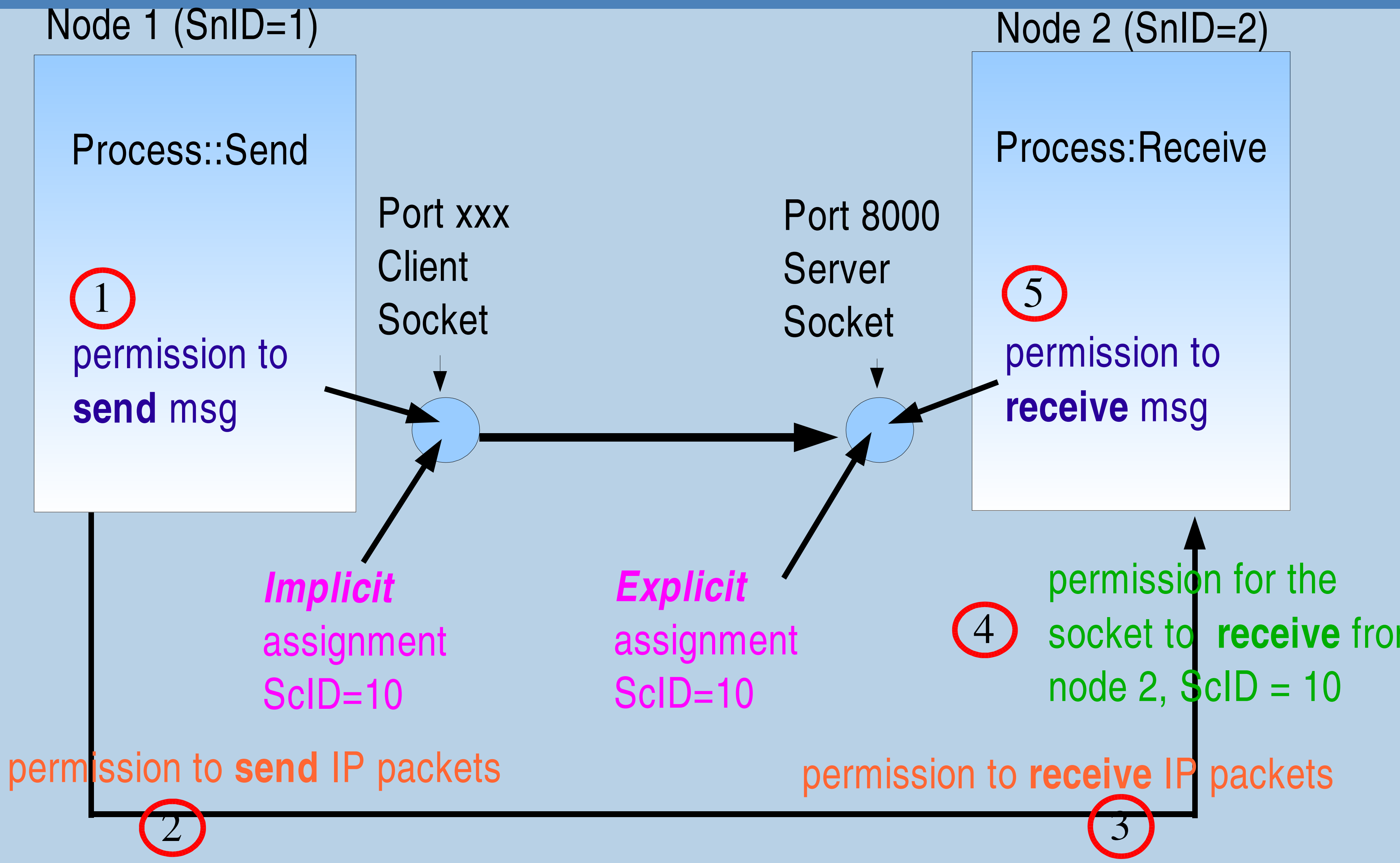
- Express a coherent security vision (security policy) through out all the cluster

- Local security policy:
  - Initially integrated to the secure boot software
  - Maintained and updated by the security server through security communication channel


Node 1 (SnID=1)
Node 2 (SnID=2)
Source Process (ScID=10)
Target Process (ScID=20)
<sScID> 10 </sScID>
<sSnID> 1 </sSnID>
<tScID> 20 </tScID>
<tSnID> 2 </tSnID>
<allow> Connect </allow>
Distributed Security Policy

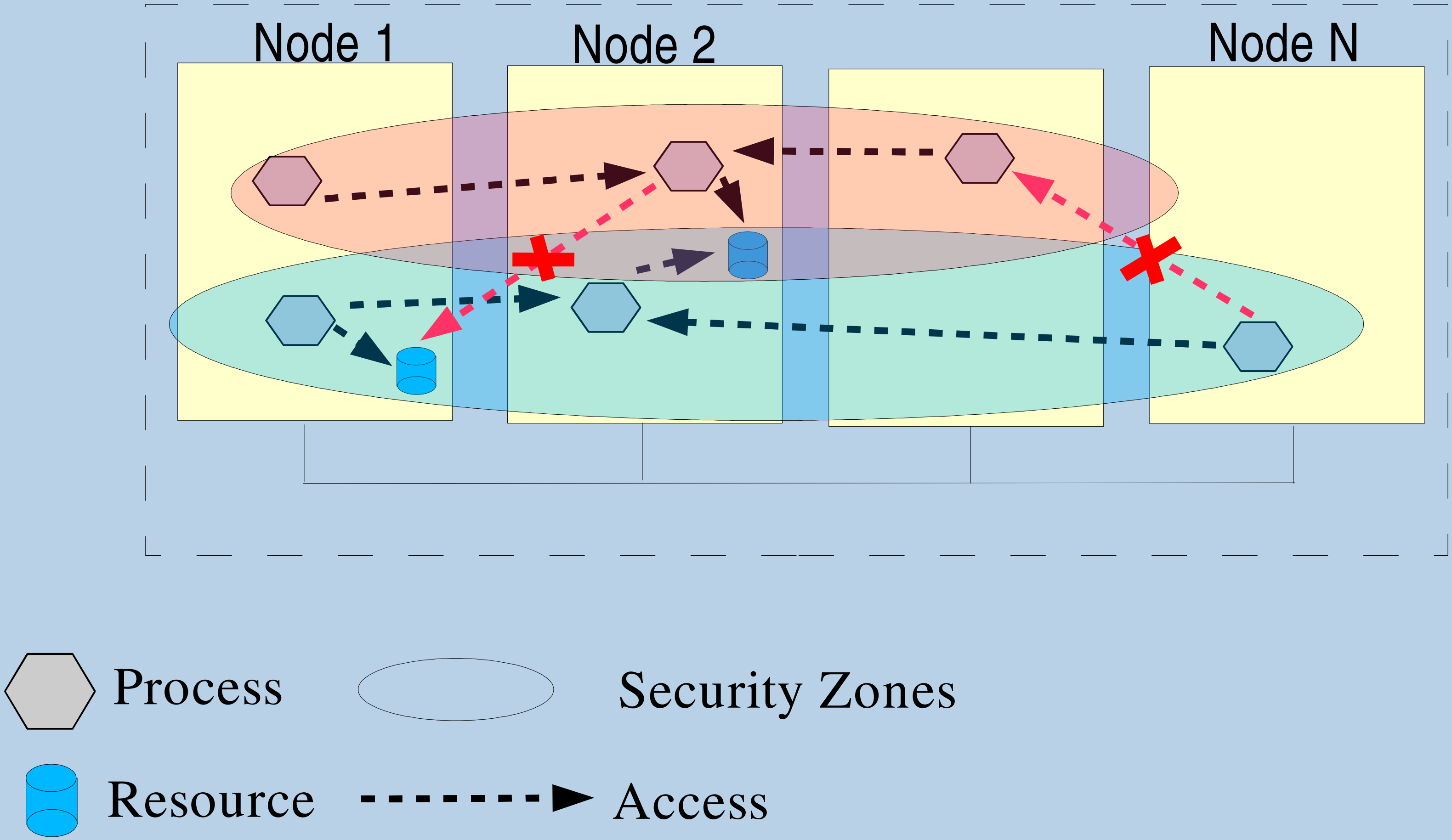
DisAC: Access control verifications




Node 1 (SnID=1)
Node 2 (SnID=2)
Process::Send
Process:Receive
Port xxx
Client
Socket
Port 8000
Server
Socket
1
permission to
send msg
5
permission to
receive msg
Implicit
assignment
ScID=10
Explicit
assignment
ScID=10
4
permission for the
socket to receive fro
node 2, ScID = 10
permission to send IP packets
permission to receive IP packets
2
3

ERICSSON

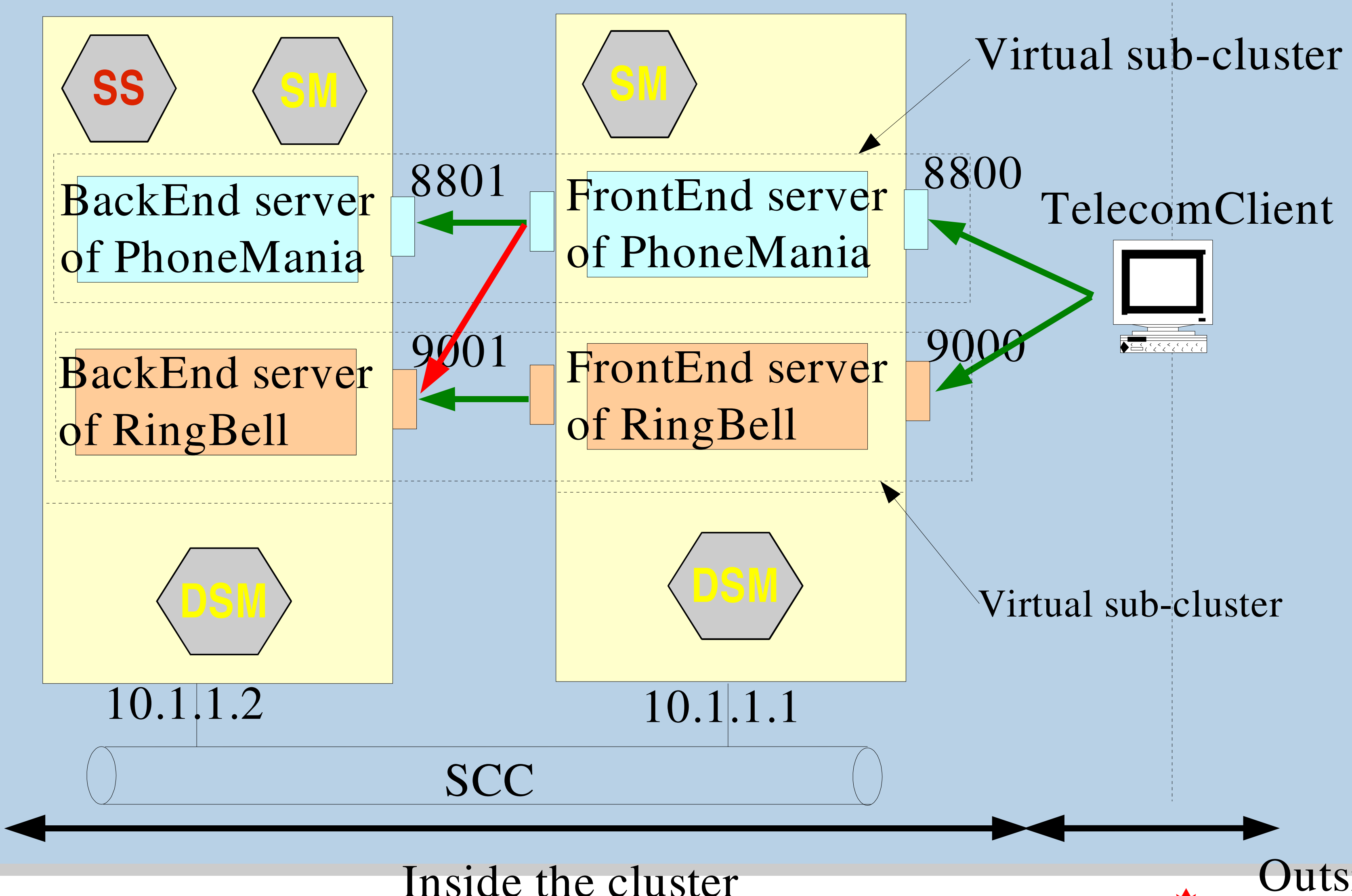
Cluster-wide Access Control



Security zones

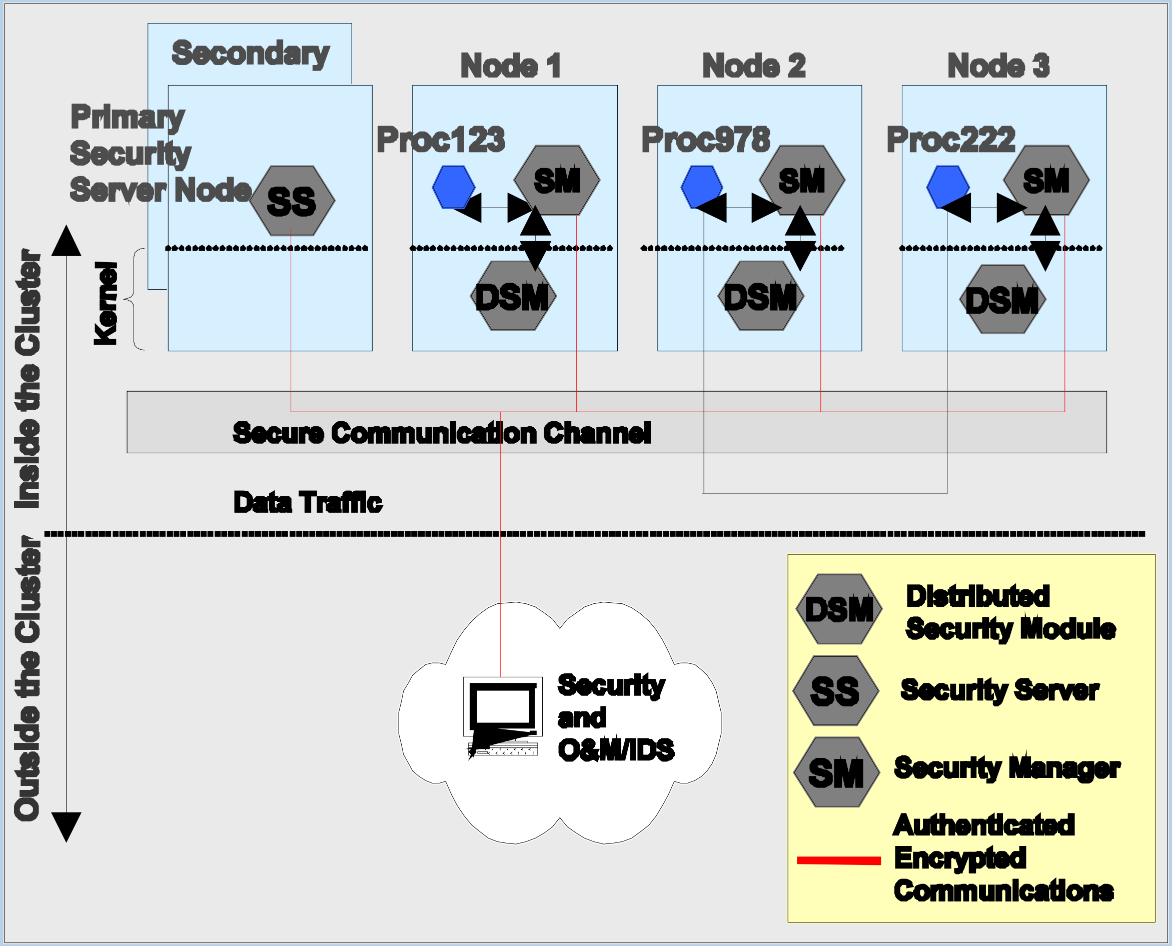
- Each ScID corresponds to a security zone (few ScIDs)
- ScIDs are not meant to be unique for each process or application
- Possible simplification by using ALL keyword in DSP for SnID allows to avoid the locality in the cluster
- Very simple; using two ScIDs and SnID set to ALL; it is possible to divide the cluster into two zones: trusted and untrusted zone

DisAC: Creating Virtual Security Zones

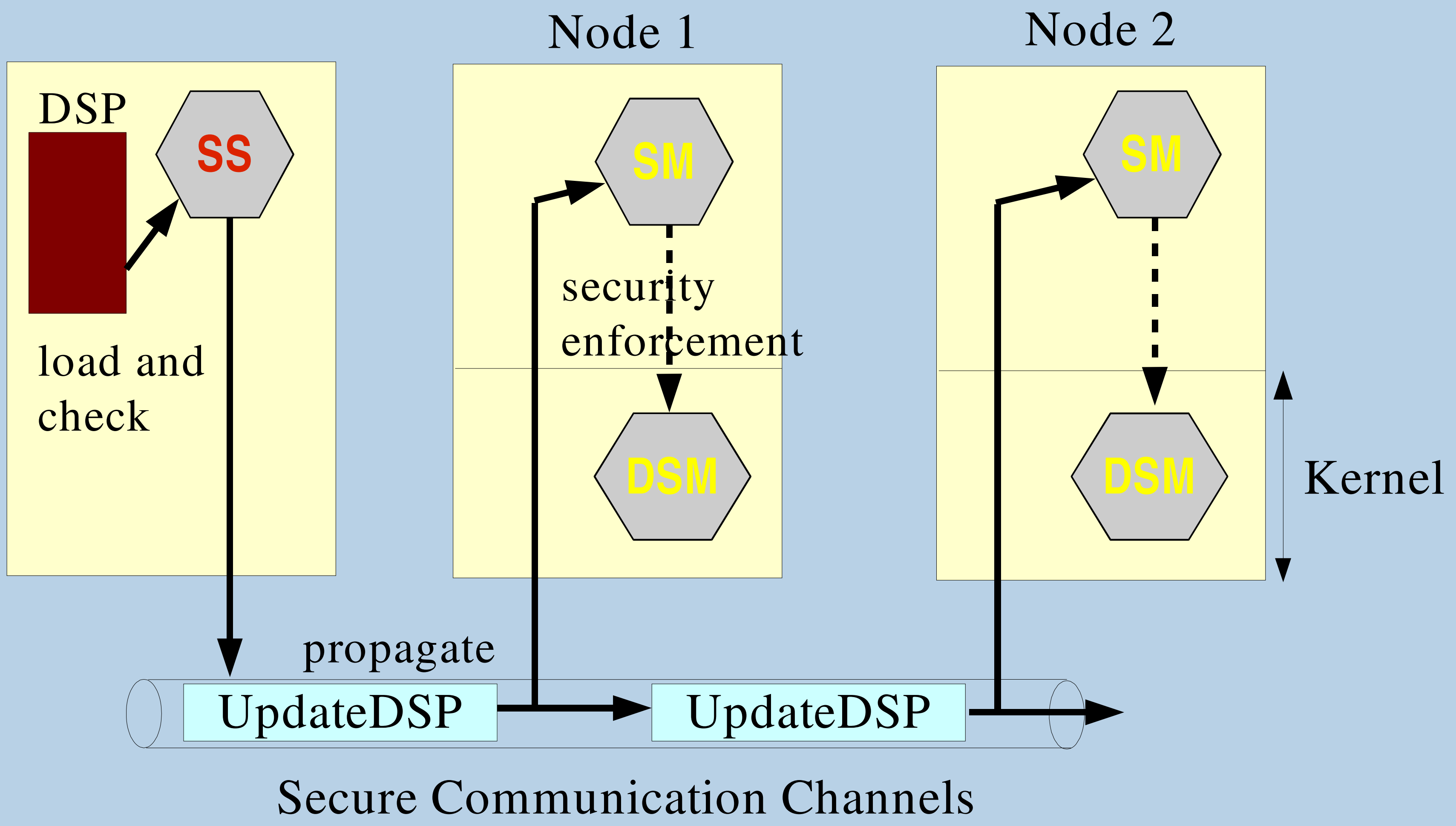


DisAC implementation:
Distributed Security Infrastructure

Distributed Security Infrastructure



DSP Update



Node 1
Node 2
DSP
SS
SM
SM
load and check
security enforcement
DSM
DSM
Kernel
propagate
UpdateDSP
UpdateDSP
Secure Communication Channels

Development Environment

- Kernel 2.4.17
- LSM patch
- Red Hat 7.3
- C/C++
- GCC 2.96

Benchmarking results

Test type	Without DSI	With DSI	Overhead
Stat	1.92	1.94	1.0%
Open/Close	2.68	2.68	0%
Fork	92.81	93.58	0.82%
Exec	322.56	328.33	1.78%
Sh proc	2140.75	2150	0.43%
UDP	9.68	10.61	9.6%
RPC/UDP	17.66	18.7	5.9%
TCP	11.08	12.68	14.4%
RPC/TCP	23.42	24.3	3.75%

Results for LMBench on Linux 2.4.17, Pentium 4, 2,4 Ghz.

Time units are microseconds.

Challenges

- Comprehensible and acceptable security policy
  - Seltzed & Schroder 1975: security mechanisms must be comprehensible and acceptable to users, or they will be ignored and bypassed.
  - Explicitly defining security zones in DSP
    - generate the low level DSP rules from those high level security zones

Conclusions

- Distributed Access Control
  - DisAC enforces cluster-wide access control at kernel level for distributed systems
  - Unified security view
  - Process level Granularity
  - Setting virtual security zones inside a cluster
- Distributed Security Infrastructure
  - Implemented under GPL (stable version for kernel 2.4.17, development version for kernel 2.6.X)

DSI References

- **Web site**

  http://disec.sourceforge.net/

  http://www.linux.ericsson.ca/dsi

- **DSI Packages**

  http://sourceforge.net/project/showfiles.php?group_id=67502

- **Online CVS**

  http://cvs.sourceforge.net/viewcvs.py/disec/

- **Contact person**

  makan.pourzandi@ericsson.com