Critical Infrastructures go to the Internet: How to Protect Them?

Nuno Ferreira Neves
Univ. of Lisboa, Portugal
http://www.di.fc.ul.pt/~nuno

>MASSIF<

Organization

- Case studies
  - Electrical Critical Information Infrastructure
  - Security Information and Event Management Systems
- Hierarchy of node protection mechanisms
- Communication protection mechanisms

Challenge on Critical Information Infrastructures (CII) Protection

- The problem of achieving resilience of CII is *relatively complex* because of the hybrid composition of these infrastructures
  - SCADA, PCS systems that yield the operational ability to supervise, acquire data and control physical processes
  - interconnections to the standard corporate intranet, where services and engineering reside
  - the Internet, to which, and often unwittingly, the SCADA network is sometimes connected to
- also because it became inter-disciplinary
  - SCADA systems are real-time systems with some reliability or fault-tolerance concern, classically not designed to be widely distributed or remotely accessed, let alone open, and designed without security in mind

Our Position

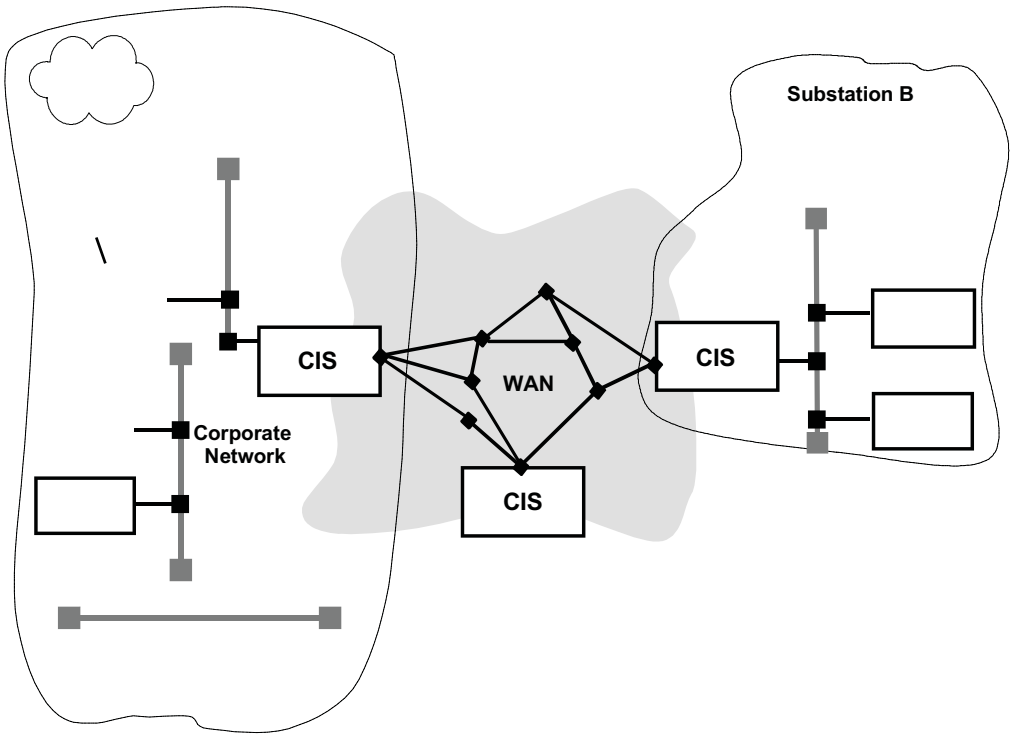
- The computer-related operation of a critical utility infrastructure became thus a *distributed systems problem*, including
  - interconnected SCADA/embedded networks, corporate intranets, and Internet/PSTN access subsystems
- and this distributed systems problem is *hard*
  - includes facets of real-time, fault-tolerance, and security
- Further insight on the CII problem
  - CII feature a lot of legacy subsystems and non-computer-standard components (controllers, sensors, actuators, etc.)
  - Conventional security and protection techniques, when directly applied to CI controlling devices, sometimes stand in the way of effective operation

How do we achieve resilient operation in CII at the architectural level?

Two ideas that can help us guide our thoughts

- Classical security and/or safety techniques alone will not solve the problem because they will put us at the level of current ICT infrastructures

- Any solution passes by automatic control of macroscopic command/information flows essentially between local/virtual LANs composing the CII



MIS Function

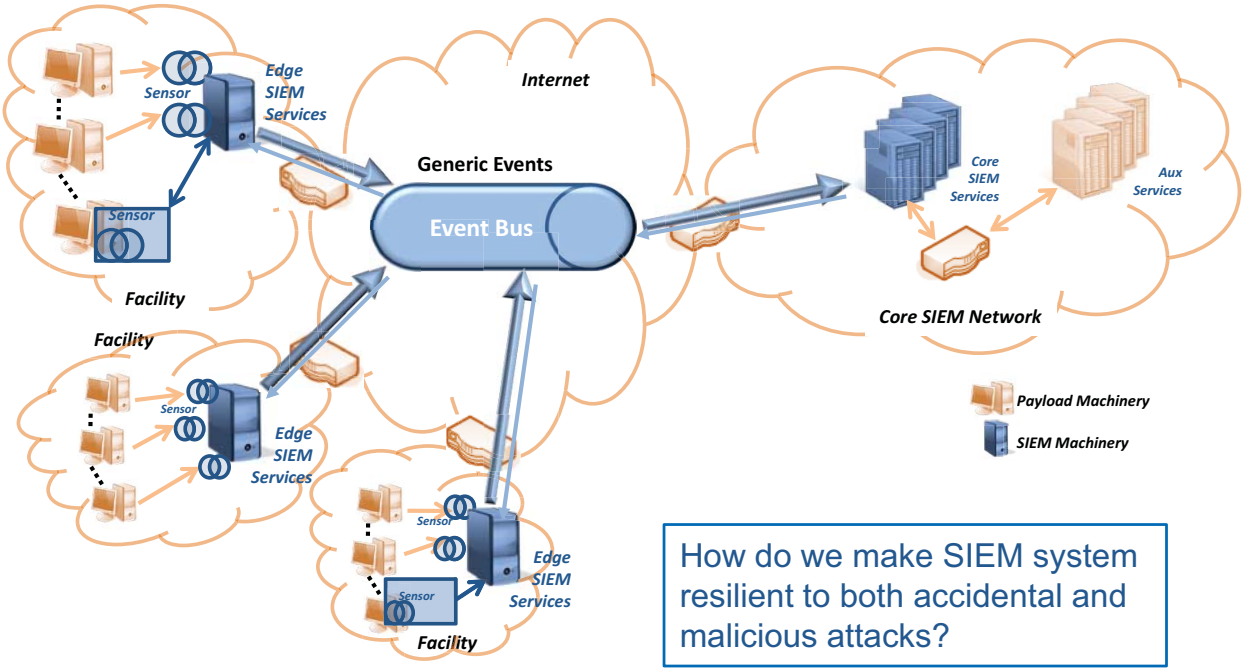
- MIS act as a set of servers providing distributed services
  - achieving *control of the command and information flow*, and securing a set of necessary *system-level properties*
  - like *sophisticated firewalls* combined with intrusion detectors, connected by distributed protocols

- Fundamental services (at application point of view)
  - Protection: ensures that the incoming and outgoing traffic to/from the LAN is in accordance to the CII security policy
  - Communication: provides dependable inter-MIS communication (point-to-point, broadcast, etc).

Security Information and Event Management (SIEM) Systems

- SIEM systems offer various capabilities for the collection and analysis of security information in networked infrastructures
  - integrating a large range of security and network tools
  - allowing the correlation of thousands of events and the reporting of attacks and intrusions in near real-time
- Main components
  - Sensors: collect information about the local environment and help on the responses; Can be: signature or anomaly-based IDS; vulnerability scanners; network profiling; inventory management
  - Collectors: gather and normalize the events generated by the sensors and any external systems; can be deployed standalone or in a Sensor
  - Management server (or SIEM core engine): event correlation and real-time monitoring; risk assessment; reporting and data mining; network profiling and inventory management

SIEM Architecture – A Topological View



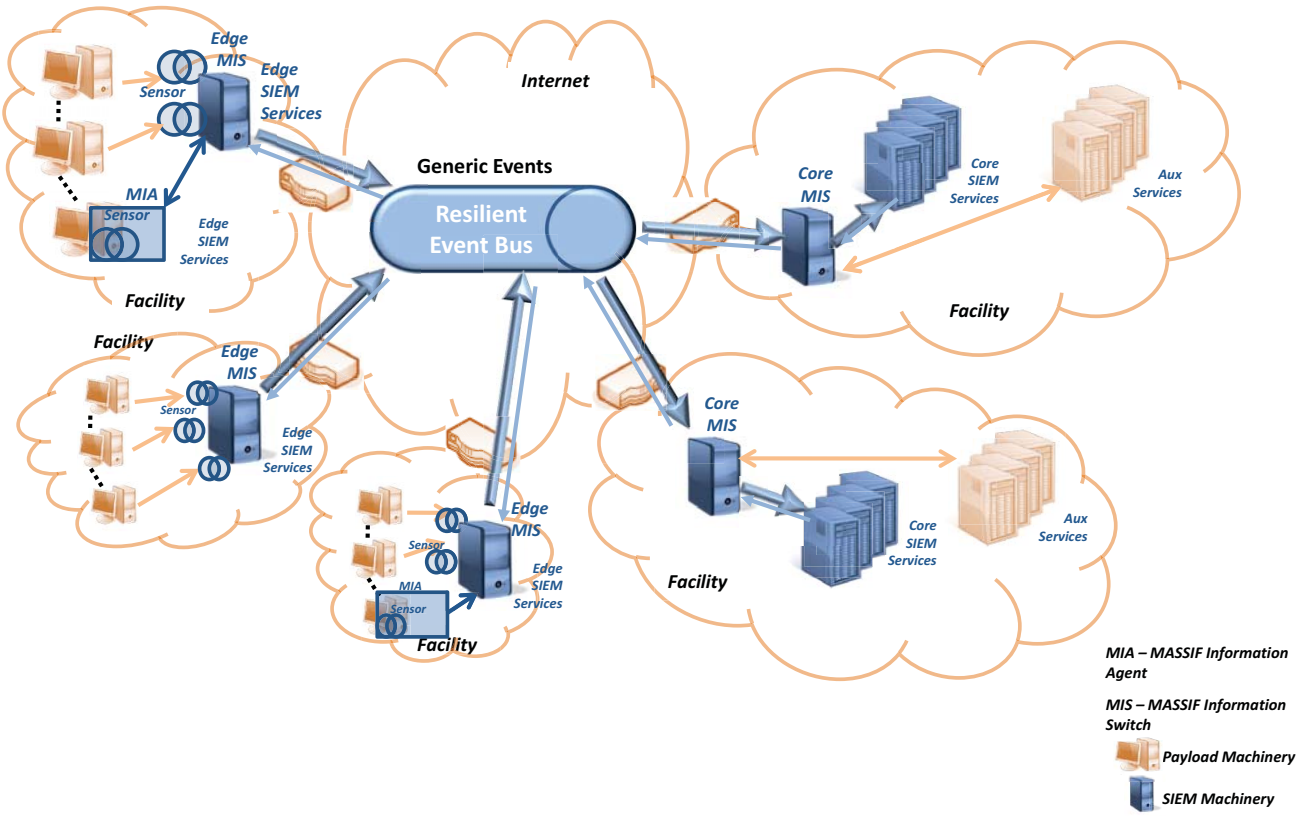
RATIONALE for an SIEM RESILIENT INFRASTRUCTURE

- Complement classical security techniques with resilience mechanisms
  - largely based on prevention, human intervention and ultimately disconnection
  - need for achieving tolerance, automation, and availability under attack
- Promote automatic control of macroscopic information flows
  - between layers of increasing resilience, from unprotected edge facilities up to the necessarily protected core processing units
- Reconcile resilience with legacy preservation
  - interfere as least as possible with the target system
  - SIEM integration should be as seamless and as transparent as possible
- Avoid single points-of-failure
  - at the edge level: protection the event collection
  - at the communication level: integrity and timeliness of the information flow
  - at the core processing level: availability and integrity of event processing
- Secure timeliness in the presence of faults and attacks
  - in different grades of real-time, from edge to core
  - detecting timing failures when timeliness enforcement is impossible

Overview of the resilient MASSIF architecture

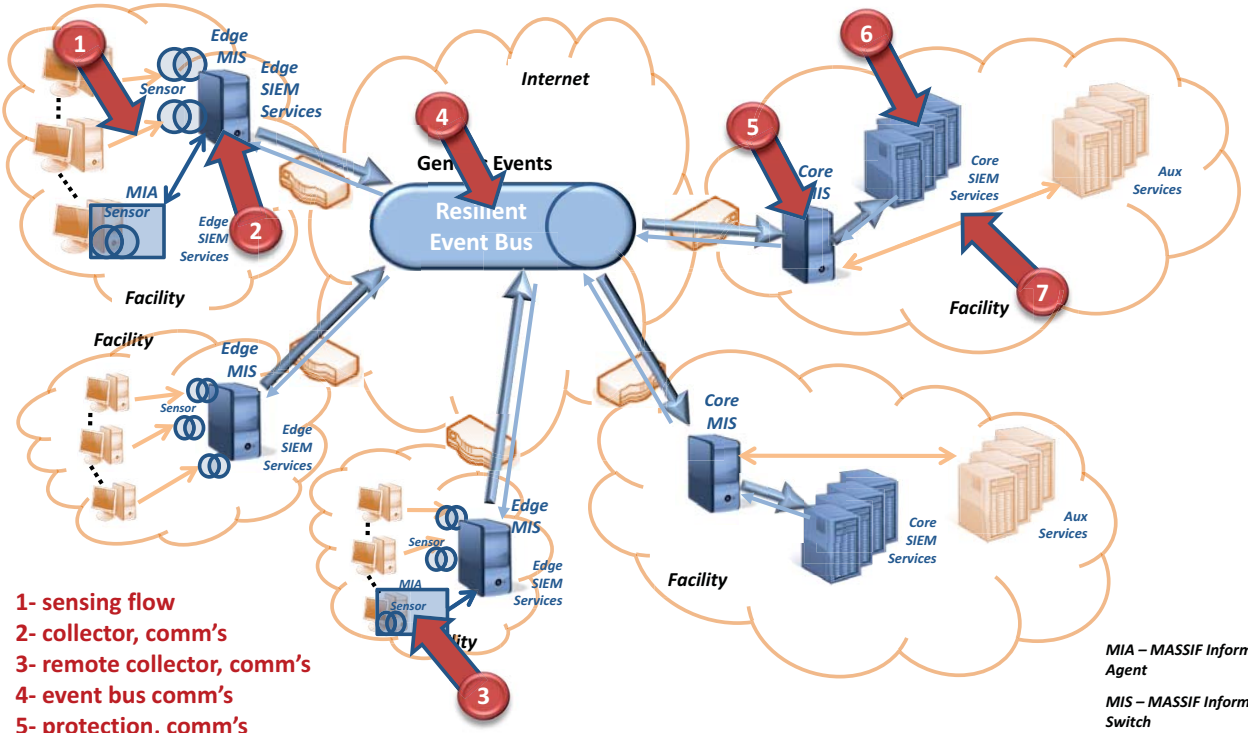
- Main characteristics
  - the architecture is laid down as a *sort of overlay* on the target, so as to preserve legacy but allow seamless integration
  - modeled pretty much as a SCADA system, producer-consumer system upstream, with low bandwidth commands downstream
- Resilience procurement based on
  - securing the information flow
  - protecting crucial processing units
  - making the dissemination infrastructure itself resilient
  - implementing all functions in a modular way around conceptual devices called MASSIF Information Switches (MIS) and MASSIF Information Agents (MIA), respectively in HW and in SW

General MASSIF Architecture



General MASSIF Architecture

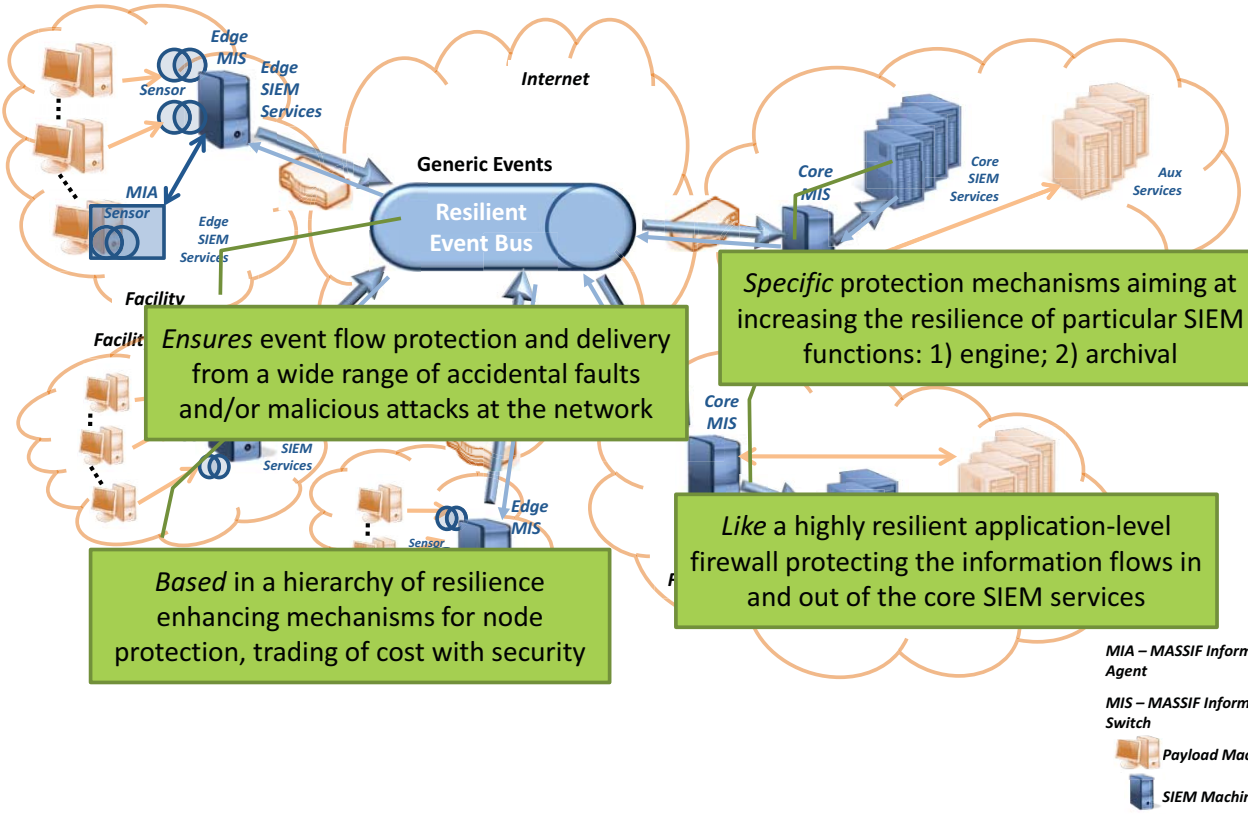
Attack Vectors



1- sensing flow
2- collector, comm's
3- remote collector, comm's
4- event bus comm's
5- protection, comm's
6- internal core systems
7- auxiliary sys. interactions

General MASSIF Architecture

Resilience solutions



Ensures event flow protection and delivery from a wide range of accidental faults and/or malicious attacks at the network

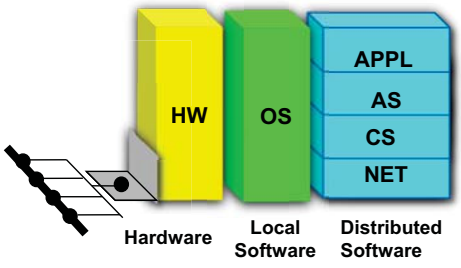
Specific protection mechanisms aiming at increasing the resilience of particular SIEM functions: 1) engine; 2) archival

Based in a hierarchy of resilience enhancing mechanisms for node protection, trading of cost with security

Like a highly resilient application-level firewall protecting the information flows in and out of the core SIEM services

HIERARCHY OF NODE PROTECTION MECHANISMS

How Do You Protect a Node?



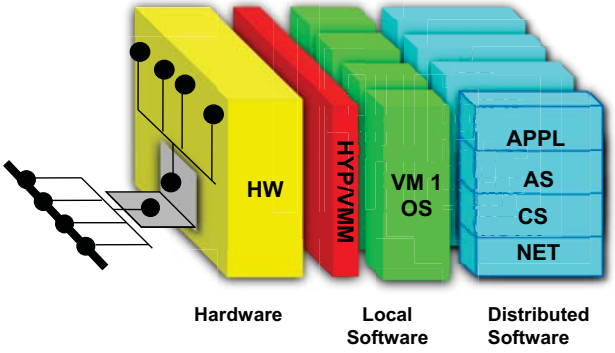
- Remove unneeded software
- Store the node in a secure room
- Diligently patch the software to the latest version
- Employ local protection software: antivirus; IDS; Firewall
- ….

- Why would you want to do more?

OS	Driver	Kernel	Sys. Soft.	App.	Total
OpenBSD	2	76	37	38	153
NetBSD	9	64	39	31	143
FreeBSD	4	153	61	61	279
OpenSolaris	0	15	9	7	31
Solaris	2	155	120	149	426
Debian	1	25	39	148	213
Ubuntu	2	22	8	58	90
RedHat	5	94	108	237	444
Windows2000	3	146	135	211	495
Windows2003	2	171	96	291	560
Windows2008	0	123	36	175	334
% Total	1.0%	33.5%	22.5%	42.9%	

NOTE: study based on NVD database for the years 1994-2011 for vulnerabilities classified as from the operating system

Replicate the Node

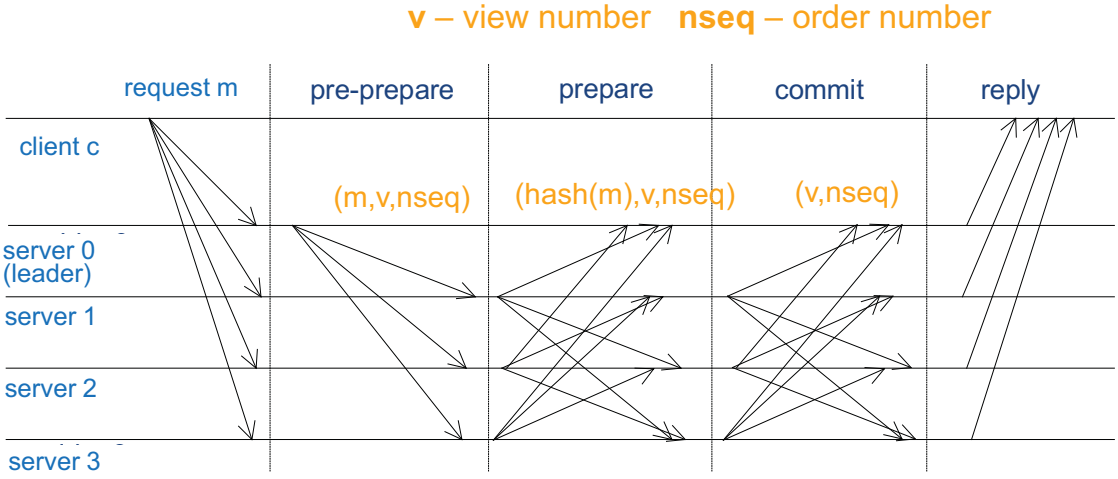


- You need to run a Byzantine Fault-Tolerant (BFT) protocol among the machines to enable correct execution even if some nodes are intruded
- Up to *f* intrusions can be tolerated in a system with $n \geq 3f+1$
- Remove time assumptions (asynchronous) but this requires you circumvent FLP impossibility result

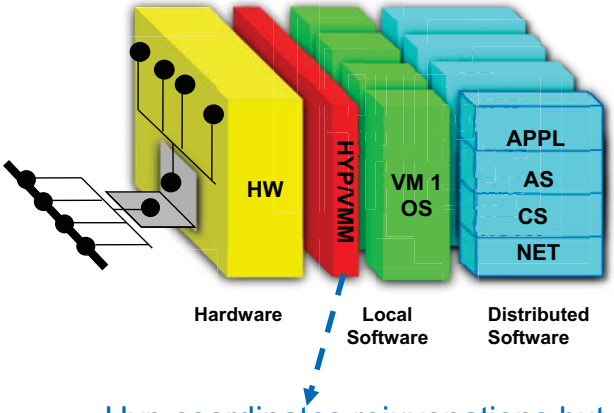
- Example BFT protocol
- Why would you want to do more?
  - given enough the adversary will be able to compromise more than f machines!!!

CL-BFT: Normal Execution

v – view number **nseq** – order number



Proactive Recovery



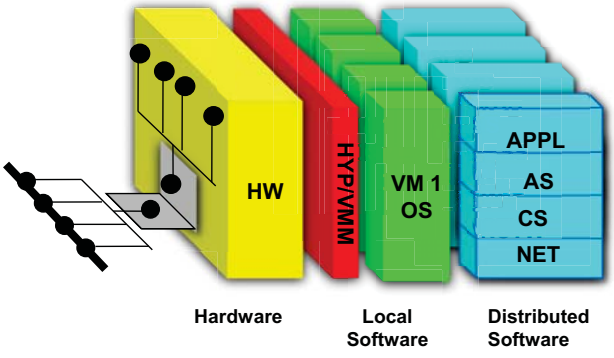
Hyp coordinates rejuvenations but needs stronger sync assumptions

- Periodically rejuvenate the replicas with a fresh operating system and application data
- As long as the rejuvenation period is small enough, the adversary cannot compromise enough replicas
- Difficulties with rejuvenation
  - needs to start at specific points in time
  - has to be performed within a limited interval of time

  which is impossible in async systems

- Why would you want to do more?
  - the adversary can use the compromised replicas to carry out malicious actions until rejuvenation is performed

Proactive-Reactive Recovery (or Self-Healing)

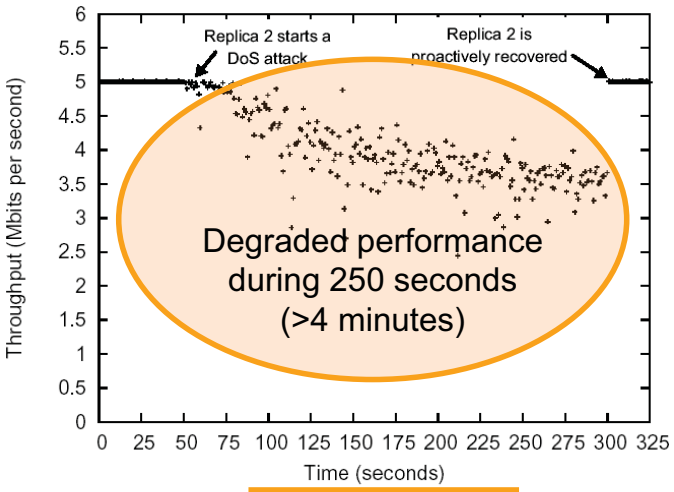


- Rejuvenate the replicas periodically and when malicious behavior is detected
- An intrusion detection mechanism is required plus the coordination of simultaneous rejuvenations
- Needs *k* extra replicas to preserve availability
- Compare PR and PR recovery

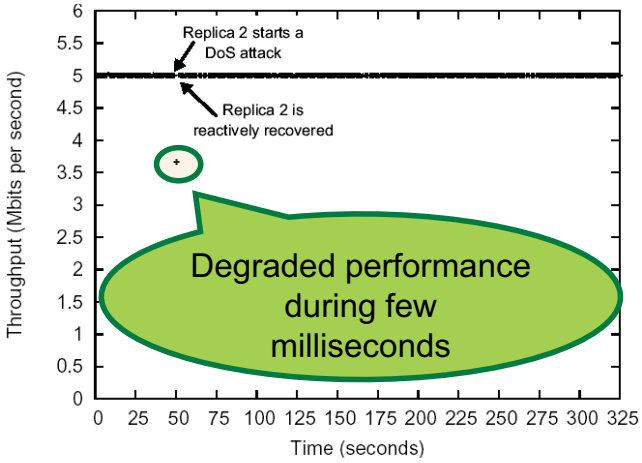
- Why would you want to do more?
  - the adversary can learn from previous attacks the vulnerabilities of the replicas and perform the attacks faster than the rejuvenation interval and without detection

Comparison between Proactive and Proactive-Reactive Recovery

- Throughput under a DoS attack from a compromised replica

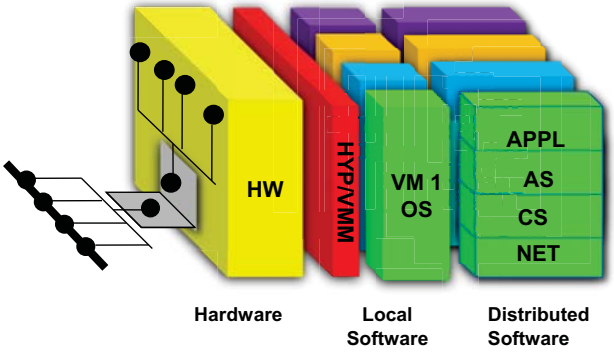


(a) With proactive recovery only.



(b) With proactive and reactive recovery

Self-Healing with Diversity



- Prevent common vulnerabilities from occurring by
  - running in each replica a different software
  - when rejuvenating a replica use a different software

- How do we get "different software" for the same functionality?
  - generate automatically multiple versions, e.g., with obfuscation techniques
  - leverage off-the-self software that provides the same functionality, e.g, at the operating system level or in specific applications (databases)

- How are we assured that they do have common vulnerabilities?

Do off-the-shelf operating systems have common vulnerabilities?

We intend to explore diversity
… among different OSes
… among different releases of the same OS
to build more resilient intrusion tolerant systems

Why Operating Systems?

- OSes play a critical role in every system
- A substantial part of the code of a replica is the OS
- People will resort to an OS rather than build their own
- There are plenty of OSes available ⇨ many options for diversity

Data Source – Raw Data

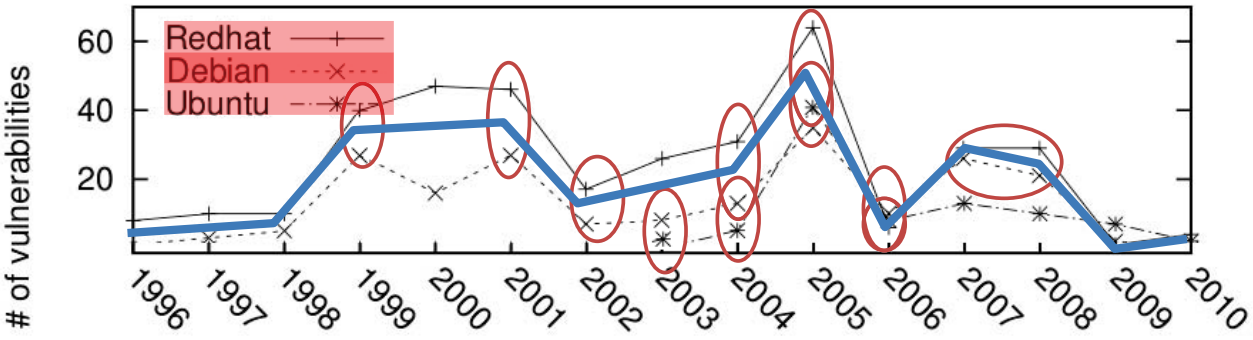
- All vulnerabilities from **National Vulnerability Database** (NVD)' XML feeds
- Vulnerability reports from 1994 to 2011

Data Source – Refining the data

*OpenBSD, NetBSD, FreeBSD, Solaris, OpenSolaris, Ubuntu, Debian, Redhat, Win2000, Win2003 and Win2008

- We selected some of the *mostly used server OS products** vulnerabilities from the XML feeds and store them in a *database*
- Next, we removed 293 vulnerabilities due to their uncertainty, vague info or disputed state

Temporal distribution of the vulnerabilities for Linux family



There is a pattern on the vulnerabilities counting

In the beginning Ubuntu was much similar to Debian

Common Vulnerabilities

	Fat Server	Thin Server	**Isolated Thin Server**
Operating Systems	**All**	**No Application**	**No App. and No Local**
OpenBSD - Solaris	13	10	6
NetBSD - FreeBSD	54	41	25
NetBSD - Ubuntu	0	0	0
OpenSolaris - Solaris	27	22	6
Win2000 – Win2003	265	120	81
Win2003 – Win2008	282	125	40

In the Isolated Thin Server OpenBSD has 62 vulns and Solaris 108 and only share 6 vulns

Most of the shared vulnerabilities are in the Kernel

Even in the Fat Server scheme NetBSD and Ubuntu have 0 common vulnerabilities

OpenSolaris has only 6 vulns, which are shared with Solaris

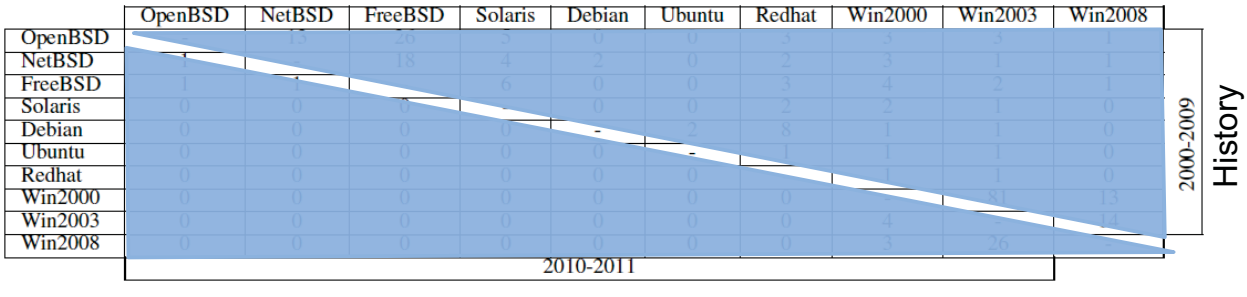
74% of Win 2003 vulnerabilities are shared with Win 2000 (up to 2010)

Most of the shared vulnerabilities are on Applications

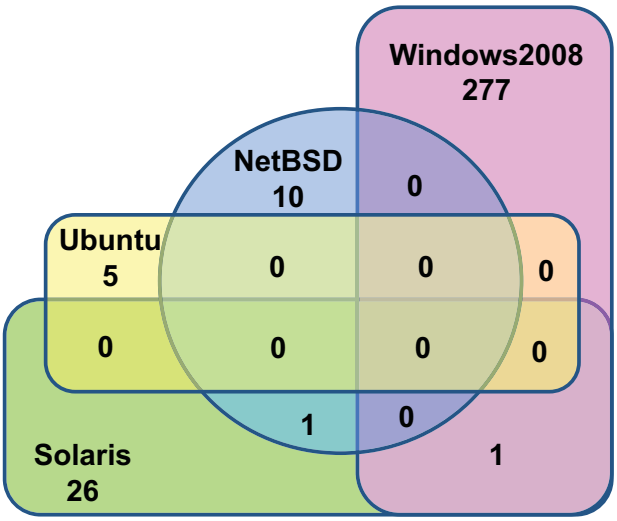
Selecting a Set with Four Diverse OSes

- There are several possible strategies to choose the diverse Oses
  - we intend to understand if we can choose OS pairs based on the past shared vulnerabilities

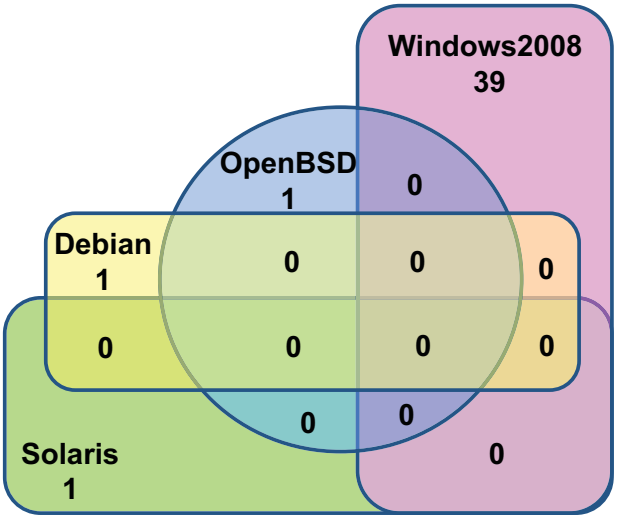
Example: Isolated Thin Servers



Example Sets (vulnerabilities 2010-2011)



Fat Servers



Isolated Thin Servers

PROTECTING THE COMMUNICATIONS

Protecting the Communications

- Secure
  - integrity and authentication
  - confidentiality
- Reliable transmission of t
  - accidental
  - malicious
- Provide some timeliness guarantees

One of the most challenging requirements because communication goes through public networks

Example Use Cases: Timeliness

- (Some) electrical sector operations with time related requirements

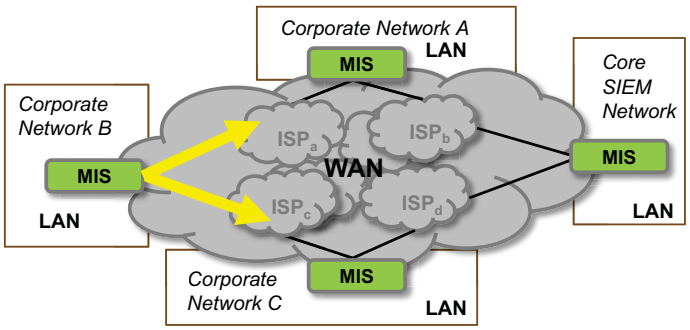
Time	Event
4s	Measurement on Human-Machine Interface
1s	Alarm signaling and switchgear status
0.2s to 2s	Teleoperation request from centres to substations
500ms	Low priority commands
200ms	High priority commands

- SIEM systems correlate security event information within a time-window (that can be as low as 1s)
- Possible approaches
  - define a uniform deadline for events based on the correlation window
  - use different deadlines depending on the source and/or type of event

Objective

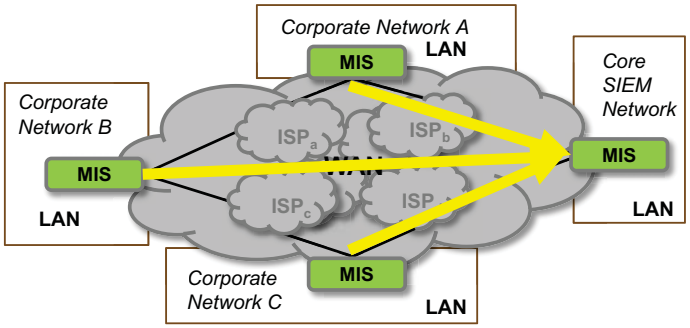
- Design a ***practical*** solution for timely and reliable communication with ***high probability*** in current CII, taking into consideration
  - compatible with current CII: should allow seamless integration without requiring major changes to the operation and organization of existing networks
  - no Internet changes: should not assume or require any special support from the underlying network, and therefore, timeliness has to be attained with best effort IP channels
  - cost consciousness: should take advantage of existing redundancy (e.g., due to over-provisioned multihoming connections), but should avoid the use of more links or costly dissemination operations (like flooding)
  - incremental integration: allow for a transition period with existing and new mechanisms

Network Considerations (1)



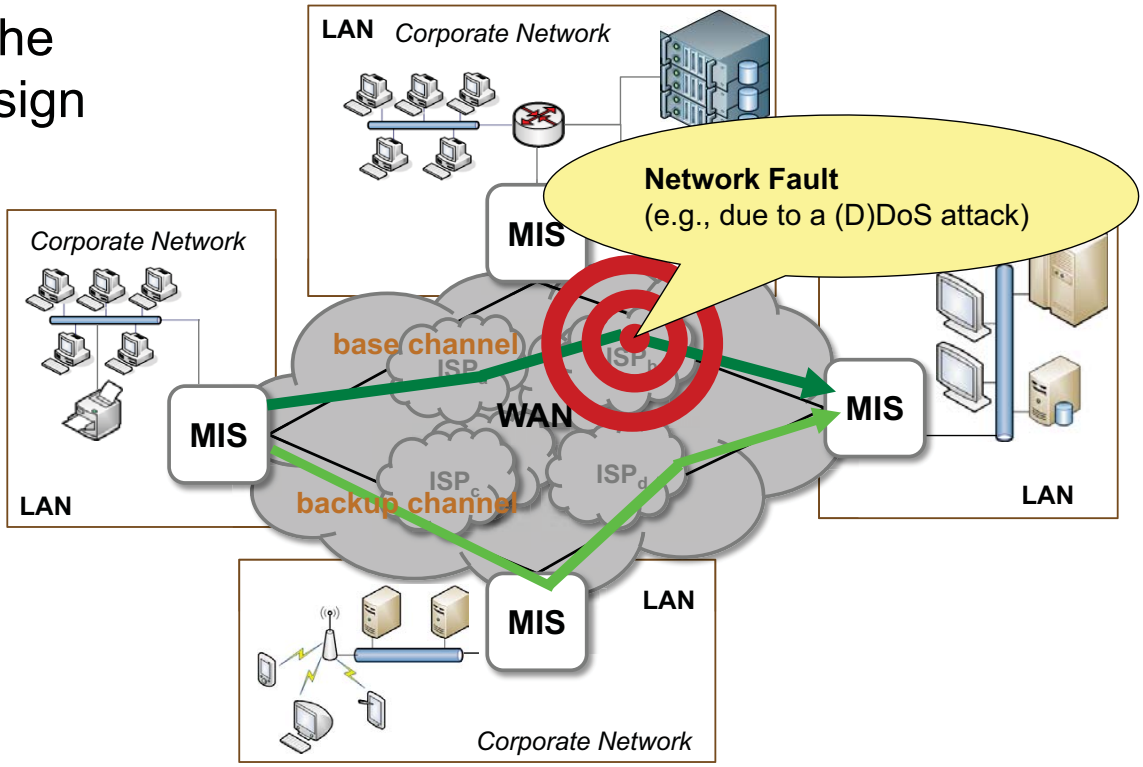
- There is a MIS at the boarder of each of the n LANs
- MIS are reliable (due to replication, self-healing, …)
- MIS are typically multihomed and are connected by a number redundant ISPs
  - contracted for link independence to get some assurances of communication availability
  - these connection can not be changed

Network Considerations (2)

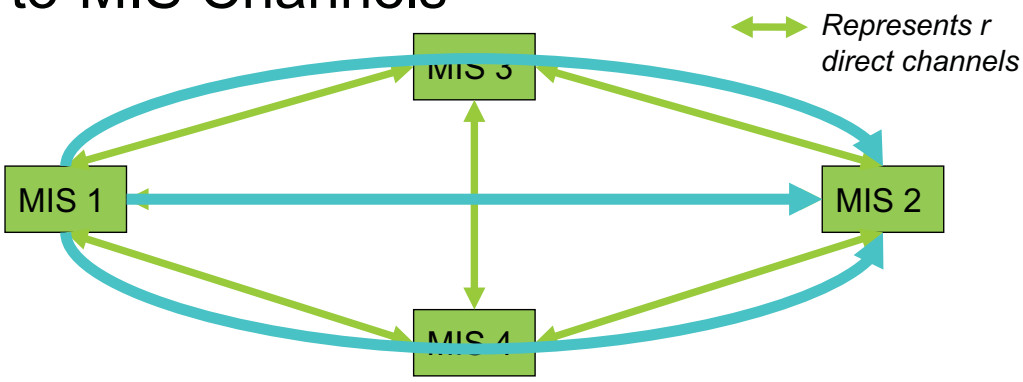


- Messages (that matter) have associated deadlines
- Closed environment with well-defined sources of legal traffic
- Mainly static
  - LANs are not added or removed too often
  - every LAN can be authenticated

Rationale of the Design



MIS-to-MIS Channels



- How many ways for a MIS to access other MIS?
  - *r* redundant ISP networks → *r direct* channels
  - *n-2* relaying MIS → $(n-2)r^2$ *indirect* channels using a **single** MIS intermediary
- Bad news
  - each MIS has $\boldsymbol{r+(n-2)r^2}$ channels to reach another MIS
  - each MIS should evaluate and choose from $\boldsymbol{(n-1)(r+(n-2)r^2)}$ channels!
  - for *r=2, n=4*: *10* channels for a single destination and a total of *30* channels
- Good news
  - several of these channels will be of little use
  - since there are many channels in a large network, we expect that some are independent

MIS Overlay Routing

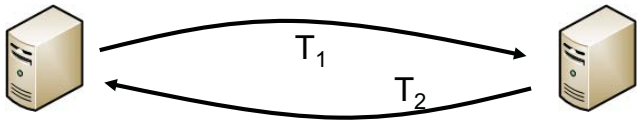
Do not attempt to minimize latency, but deliver messages just-in-time

- Fundamental ideas
  - One-hop source routing
  - Base channel + backup channels
  - Allow for a maximum number of retransmissions over diverse channels while ensuring the deadlines
- Three components of the solution
  - Measurements
  - Number of tries
  - Transmission strategy

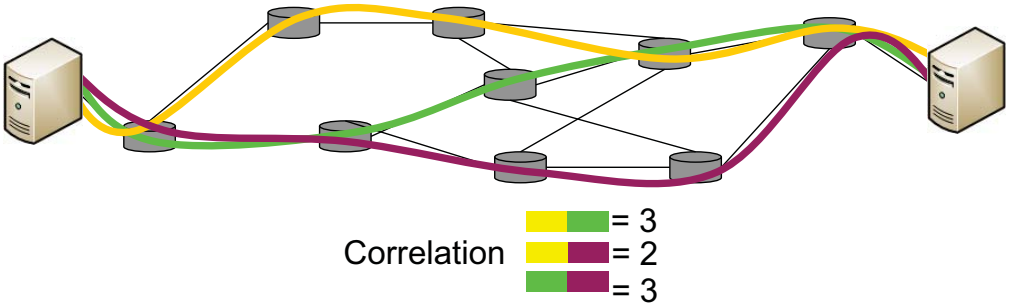
Basic Measurements (1)

- Top level routing decisions are mainly based on the *latency* among MIS nodes and on *spatial redundancy*
  - Transmission time (TXT) between two MIS

$TXT = (T_1 + T_2)/2$



  - Channel correlation: number of routers shared by two channels

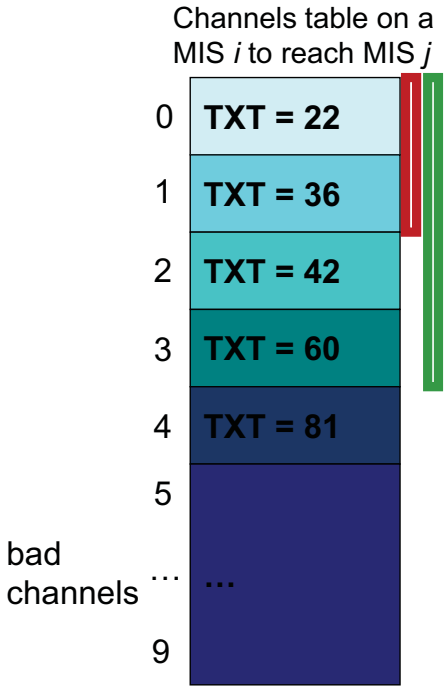


Basic Measurements (2)

- Tools
  - for TXT use `ping` and/or piggybacking
  - for correlation use `traceroute` or `netdiff`
- Other possible measurements
  - *Deadlines possibly missed (DPM)*, i.e., ACK timeout
  - Less important: *Bandwidth* and *Throughput*
- How often?
  - TXT: hundreds of seconds or few minutes
  - Correlation: minutes or hours
  - DPM: always counting…

Too many measurement can affect the network behavior

Example of the Number of Tries



m, deadline = 100

One try: 22 < 100
Two tries: 2*36+22 = 94 <100
Three tries: 2*42+2*36+22 = 178 > 100

Tries = 2

m', deadline = 300

One try: 22 < 300
Two tries: 2*36+22 = 94 <300
Three tries: 84+72+22 = 178 < 300
Four tries: 120+84+72+22 = 298 < 300
Five tries: 162+120+84+72+22 = 450 > 300

Tries = 4

Overlay Routing Strategy

- We can determine the number of tries (a maximum number of retransmissions) to send a message
- **Main Idea:** in each try, use two types of channels to send a message
  - **Base channel**: the worst channel *c* that still allows the node to try the maximum number of faster channels if it fails

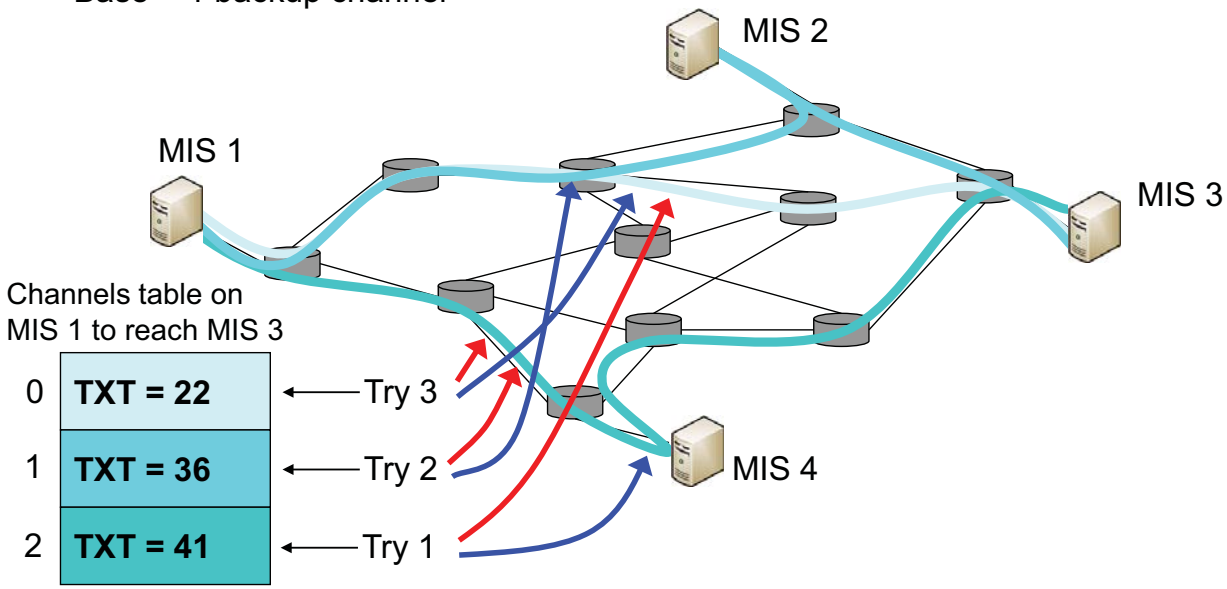
    messages don't need to reach their destination fast, they need to arrive on time!

  - **Backup channel(s)**: some other *B* channels that can deliver the message on time and that have minimal correlation with the chosen calm channel

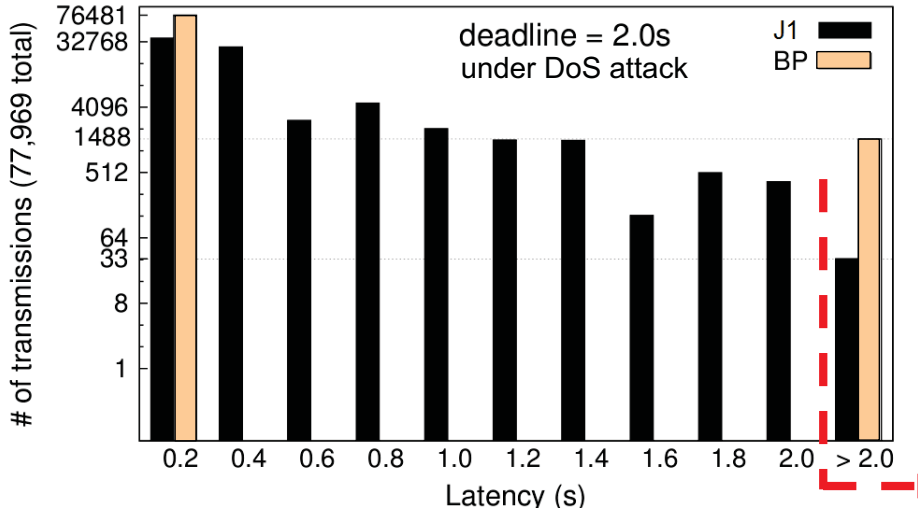
    IP networks offers no guarantee so we must take some preventive measures and use channel diversity

Example of the Overlay Strategy

- System with a single ISP connecting each MIS
- Message with deadline 180
- Base + 1 backup channel



Why not going through fastest channel first?



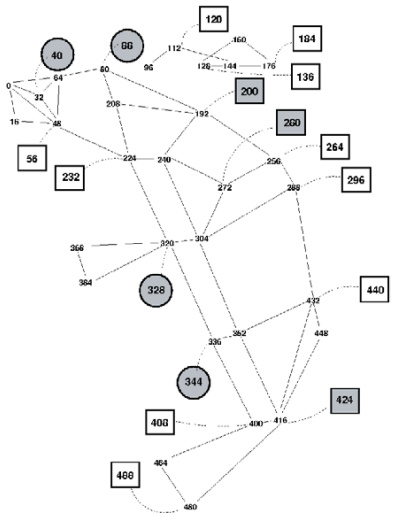
J1 – our strategy with
Base + 1 Backup
BP – best path first

Number of deadline violations

- Ends up missing much less deadlines (33 instead of 1488)
- Leaves faster channels for messages with tighter deadlines
- Achieves load balancing across the links

Evaluation

- Uses a simulation model using the J-Sim network simulation tool
- Topology based on 31 routers and 51 direct channels
- The network is duplicated to simulate multihoming
- Links with 50 ms of latency and bandwidth of 1Gbps
- There are aperiodic and periodic messages with deadlines of 1, 2 and 4 seconds

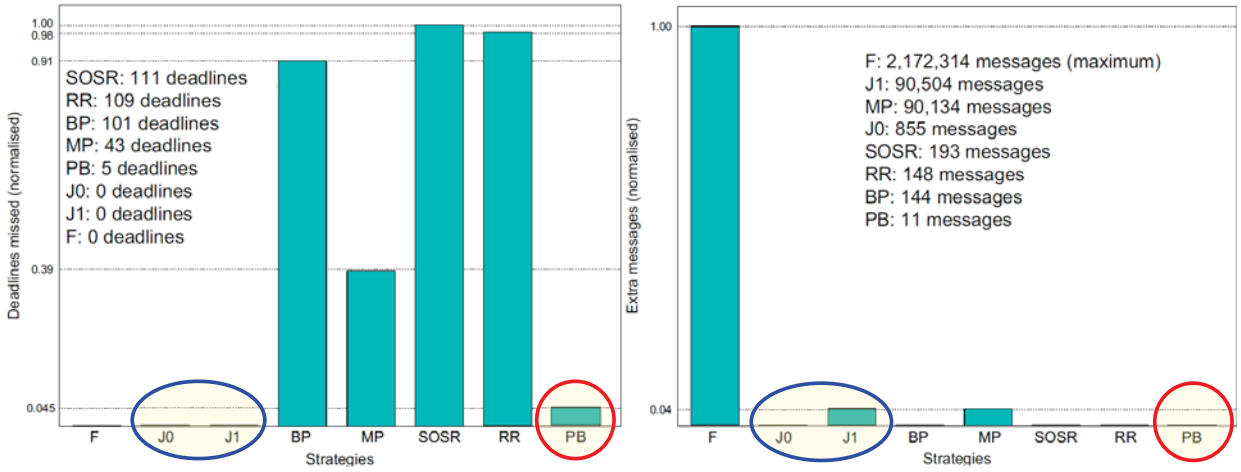


- Faults are injected following several models reported in the literature
  - in each run, 74 faults are injected in each ISP backbone
  - both accidental and DoS are modeled

Strategies under Evaluation

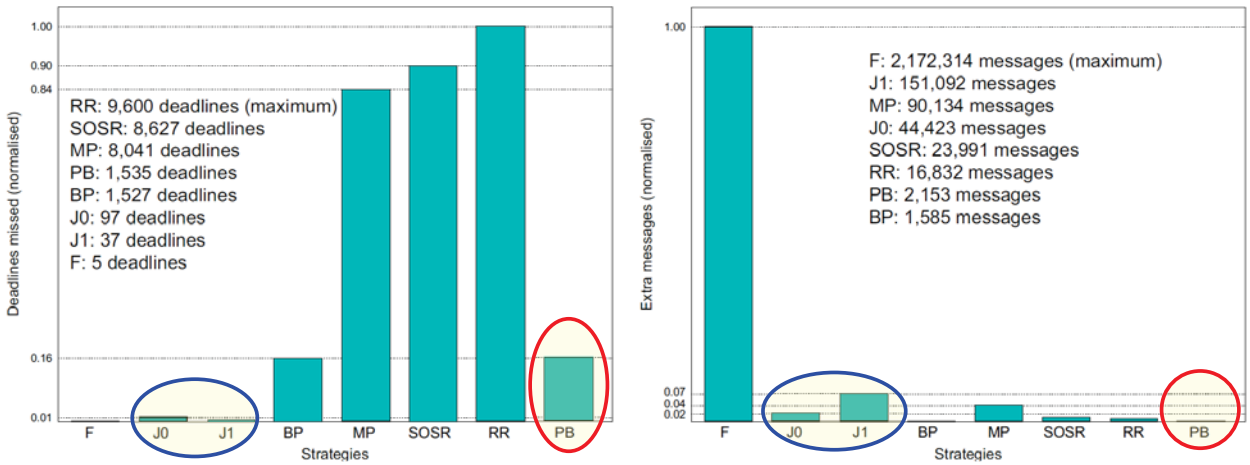
- **J0**: only the base channel
- **J1**: base + 1 backup channel
- **Flooding (F)**: all messages are sent to all channels
  (most reliable solution… uses all redundancy available)
- Multihoming (two channels):
  - **Round-robin (RR)**: direct channels used in round robin fashion
  - **Primary-backup (PB)**: if some direct channel fails, use another
- Overlay (at most one relay node):
  - **Best path (BP)**: always sent through the best non-failed channel
  - **Multi path (MP)**: send through the direct channel and a randomly selected channel (direct or not)
  - **Hybrid (SOSR)**: send first in a random direct channel, then if there is a failure use 4 random channels (direct or not)

Evaluation: Accidental Faults



- Our strategy misses no deadlines requiring only 4% of the extra messages used in flooding
- Primary-backup (as used today in utility companies) is sufficient to deal with most accidental faults (only misses a few deadlines)

Evaluation: Malicious Attacks



- Flooding misses 5 deadlines
- Our strategy misses only 37 (or 97) deadlines, using less than 7% of the extra messages used in flooding
- Primary-backup misses a significant number of deadlines (1535)

Summary

- We have looked into the problem of increasing the resilience of
  - Electrical CII
  - SIEM Systems
- Looked at different mechanisms to increase node resilience
- Described ways to improve the communication subsystems

Thank you!

Questions?