Models for Scalable and Fault Tolerant Mobile Code

Gian Pietro Picco
Dipartimento di Elettronica e Informazione
Politecnico di Milano, Italy
`picco@elet.polimi.it`
`http://www.elet.polimi.it/~picco`

Solving Problems or Leveraging Assets?

- Scalability and fault tolerance are the defining problems of contemporary network-based applications. Nevertheless:
  - on one hand, this still holds applications exploiting mobile code
  - on the other hand, ***scalability and fault tolerance are often claimed as the* raison d'être *of mobile code***
- Hence, scalability and fault tolerance should ***not*** be regarded just as a technical impedment

Design Issues: Technology and Applications

- Scalability and fault tolerance call for proper design at two different abstraction levels:
    - ➤ ***Technology:*** how to design general-purpose mechanisms for mobile code systems to help cope with scale and faults?
    - ➤ ***Application:*** how to identify the best architectural solution to meet scalability and fault tolerance requirements for a given problem?
- The two perspectives are tightly related: poorly designed mobile code mechanisms can severely impair the architectural tradeoffs

A Brief Remark on Mobile Code

- ***Observation:*** Research focuses on mobile agents systems, theories, and applications
- ***Reaction:***
  - from both a theoretical and practical standpoint, mobile agents specialize the notion of mobile code, and not vice versa
  - Hence, systems should not be designed thinking of mobile agents as the fundamental abstraction
  - mobile code is just a new tool to design distributed systems: the technology should be designed to complement rather than substitute the existing one

Scalability and Mobility

- Mobile code systems ***can*** be regarded as a solution to the overgrowth of the Internet
- But … isn't "scalable" different from "It Must Run Over the Internet"?
  - Intranet application domains for mobile code: workflow applications, network management applications, distributed simulations, …
- So … "scalable" with respect to what?
  - network traffic overhead, resource allocation, latency, response time, …
- ***How to characterize a notion of scalability?***

Politecnico
di Milano

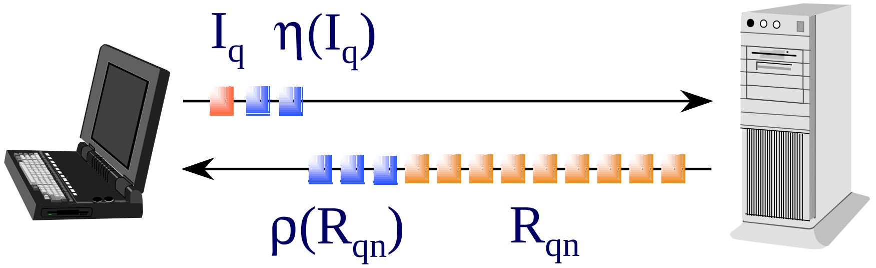
Fault Tolerance and Mobility

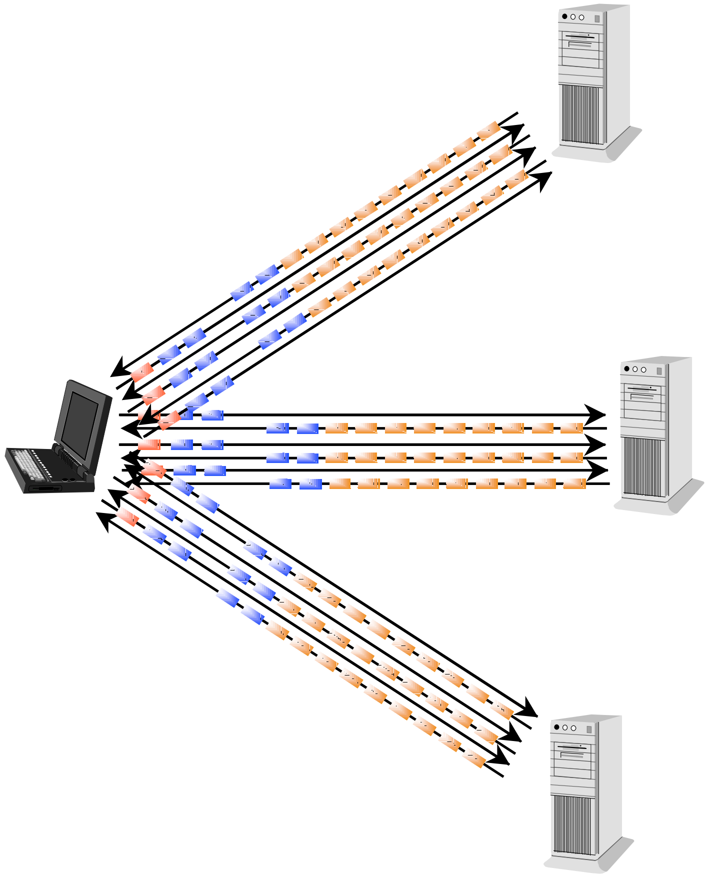
- At the application level, mobile code can improve reliability by:
  - concentrating interactions locally, thus easening monitoring and recovery
  - relocating computations based on reliability concerns
  - bypassing lossy links
- At the technological level, what are the challenges?
  - Mobility impairs reliable communication among mobile agents even on a fault-free network
- ***How to characterize the proper fault model?***

Nothing Comes for Free

- Scalability and fault tolerance often pose conflicting requirements on the designer
- Example: message delivery
  - ➤ Scalability concerns call for a compact, lightweight encoding of messages, to reduce bandwidth usage
  - ➤ Fault tolerance protocols augment message size and use more complex communication patterns, often adding network overhead
- ***How to evaluate the tradeoffs?***

Conventional Network Management





number of nodes

number of instructions

protocol overhead

$$T_{CS} = \sum_{n=1}^{N}\sum_{q=1}^{Q}\left(\eta_{CS} I_q + \rho_{CS} R_{qn}\right)$$

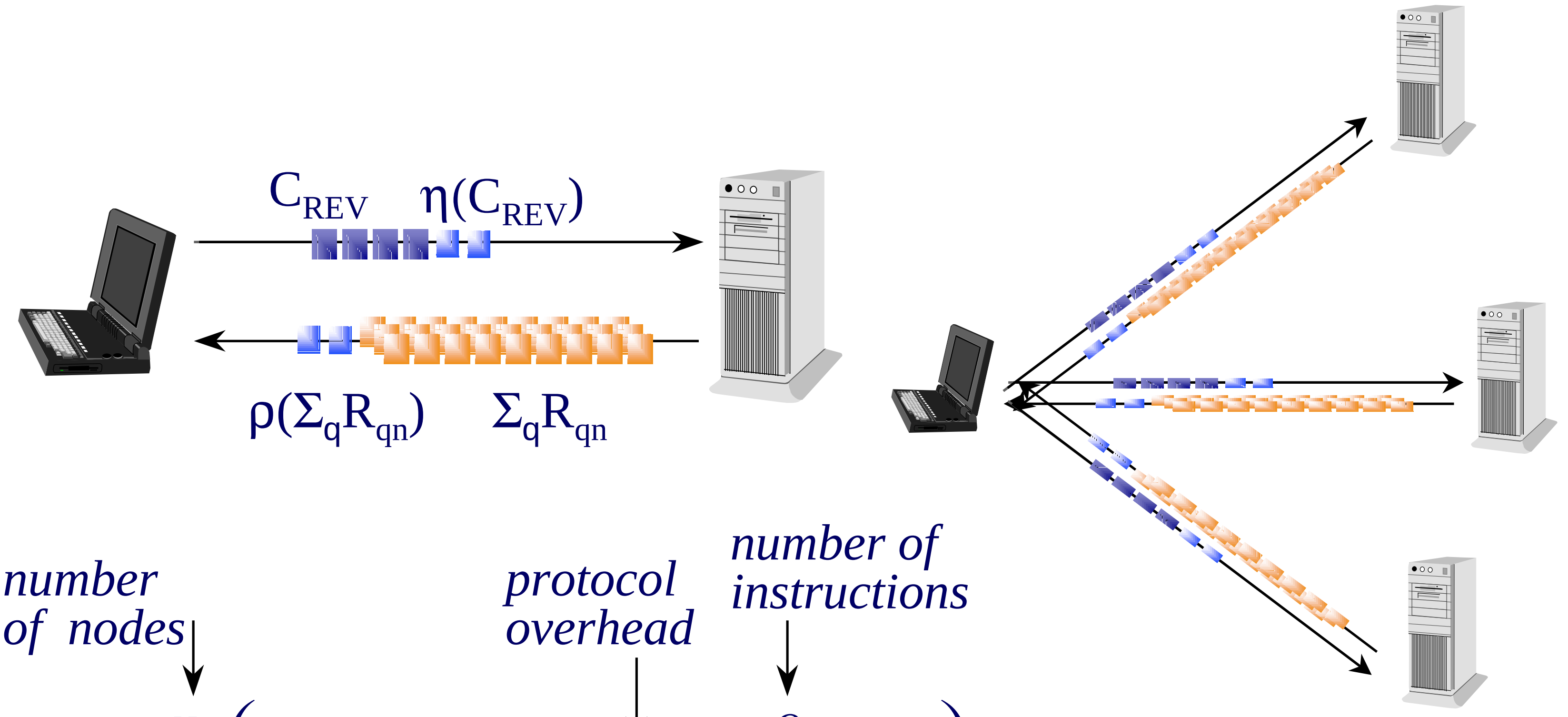
network traffic

protocol overhead

size of an instruction

size of a reply

Network Management Using Remote Evaluation



number of nodes

protocol overhead

number of instructions

$$T_{REV} = \sum_{n=1}^{N}\left(\eta_{REV}\, C_{REV} + \rho_{REV}\sum_{q=1}^{Q} R_{qn}\right)$$

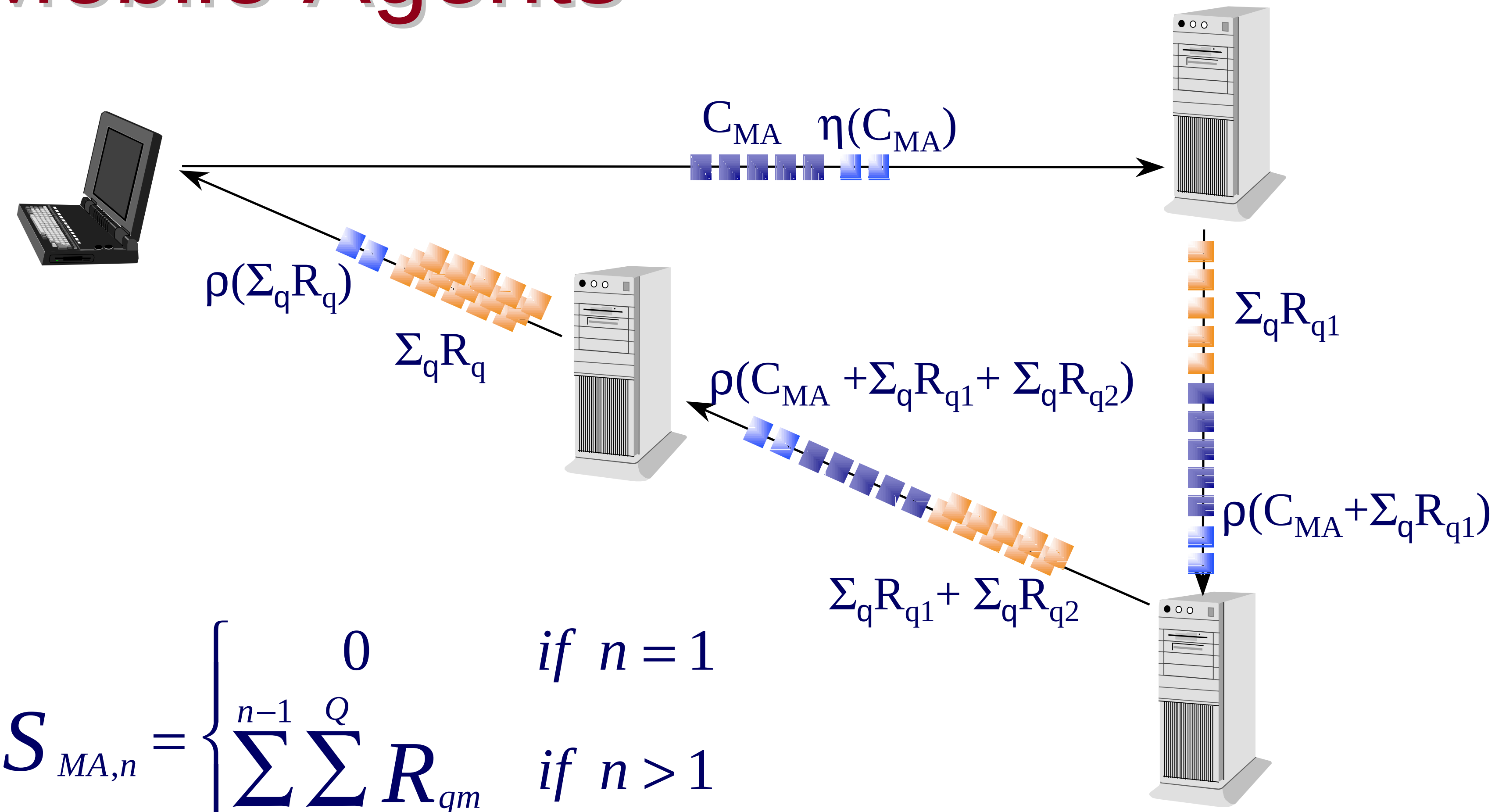
network traffic

protocol overhead

size of the code sent

size of a reply

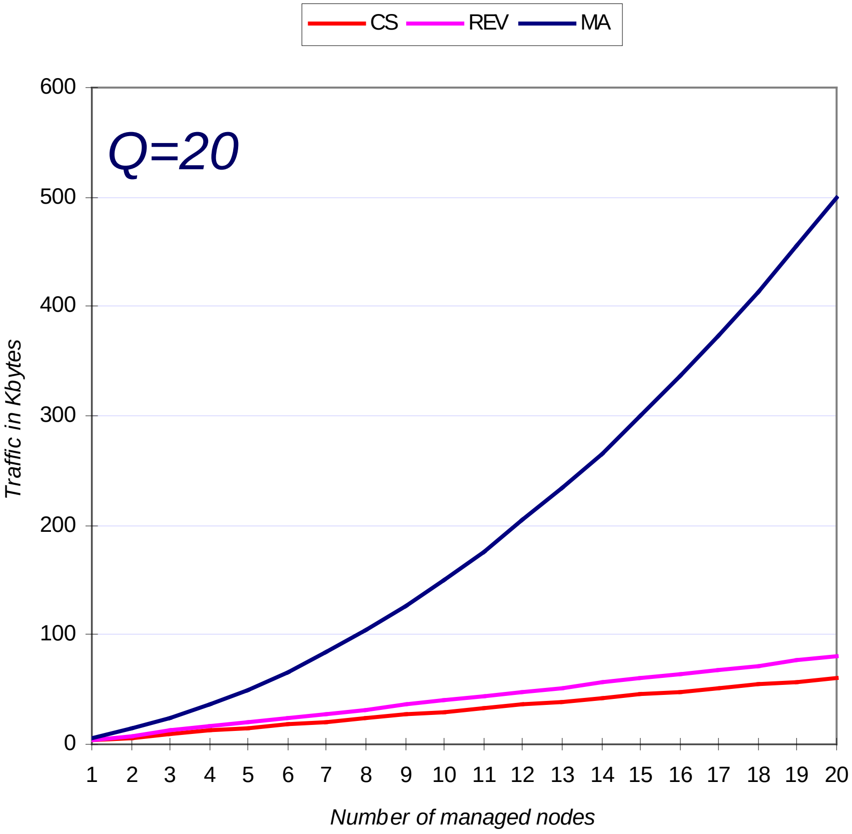
Network Management Using Mobile Agents

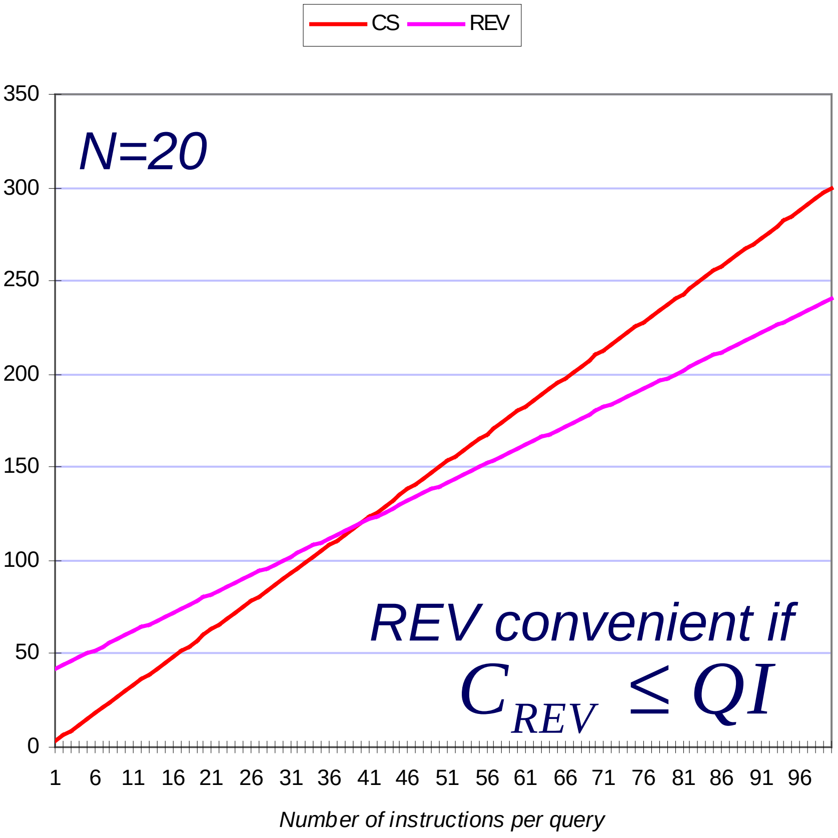


$$S_{MA,n} = \begin{cases} 0 & if \;\; n = 1 \\ \sum_{m=1}^{n-1} \sum_{q=1}^{Q} R_{qm} & if \;\; n > 1 \end{cases}$$

$$T_{MA} = \sum_{n=1}^{N} \eta_{MA} \left(C_{MA} + S_{MA,n} \right) + \rho_{MA} \sum_{n=1}^{N} \sum_{q=1}^{Q} R_{qn}$$

Analyzing Design Tradeoffs



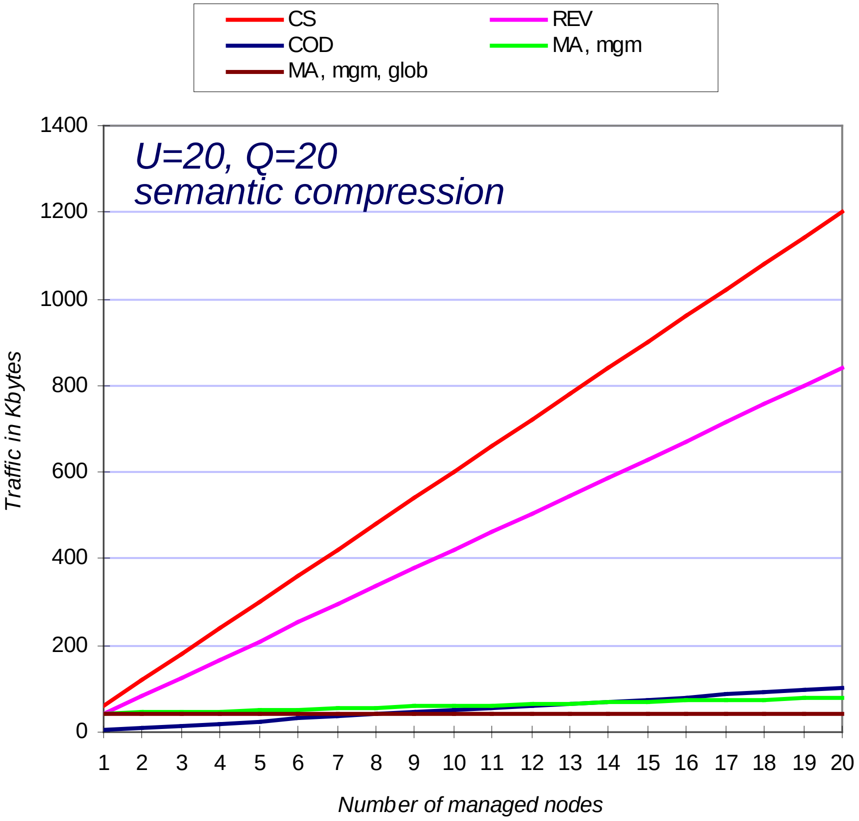


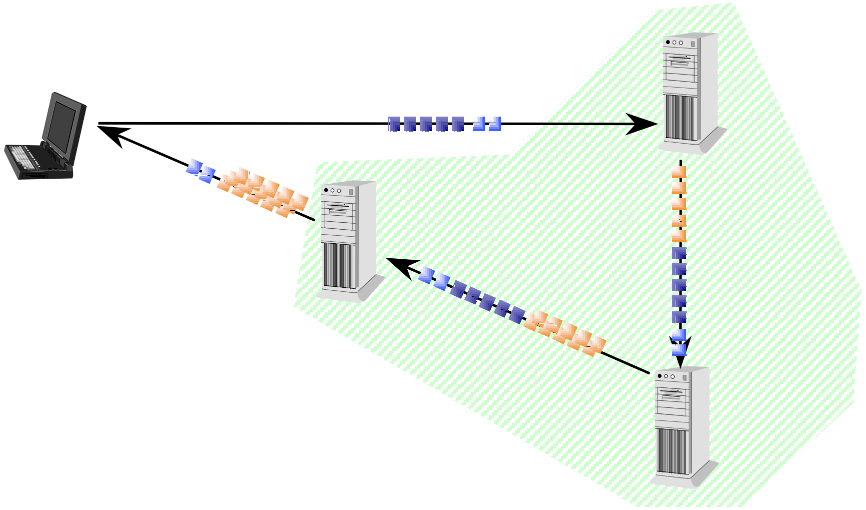
REV convenient if

$$C_{REV} \leq QI$$

Common SNMP values: I=50 bytes, R=100 bytes
Code size: C=2 Kbytes for all the mobile code paradigms
No overhead contribution (i.e. $\eta=\rho=1$) and no semantic compression

Semantic Compression and Decentralization





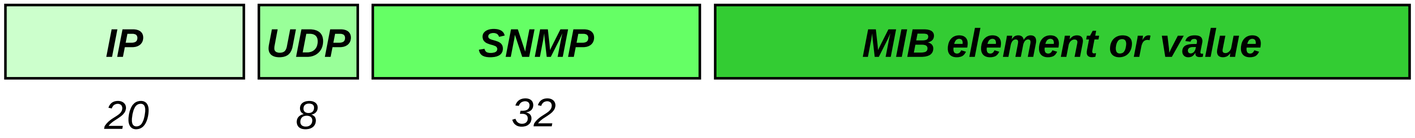
Common SNMP values: I=50 bytes, R=100 bytes
Code size: C=2 Kbytes for all the mobile code paradigms
No overhead contribution (i.e. $\eta=\rho=1$)

How Much Detail Is Needed?

- Reasoning at the architectural level gives a first idea of which style of relocation is more effective
- However, what is the impact of the mechanisms used to implement the selected architecture, as implemented in the underlying mobile code support?

Modeling the Technology

SNMP

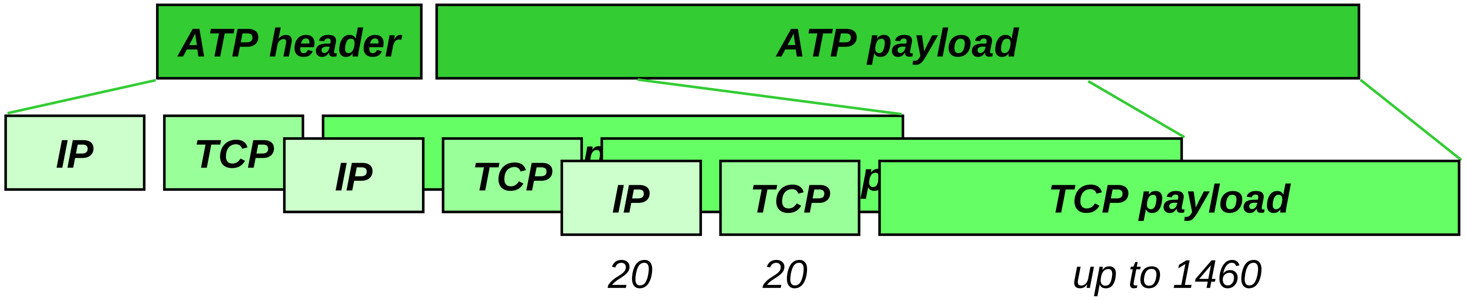


IP | UDP | SNMP | MIB element or value

20 | 8 | 32

$$\eta(X)X = 60 + X$$

Java Aglets



ATP header | ATP payload

IP | TCP | TCP payload

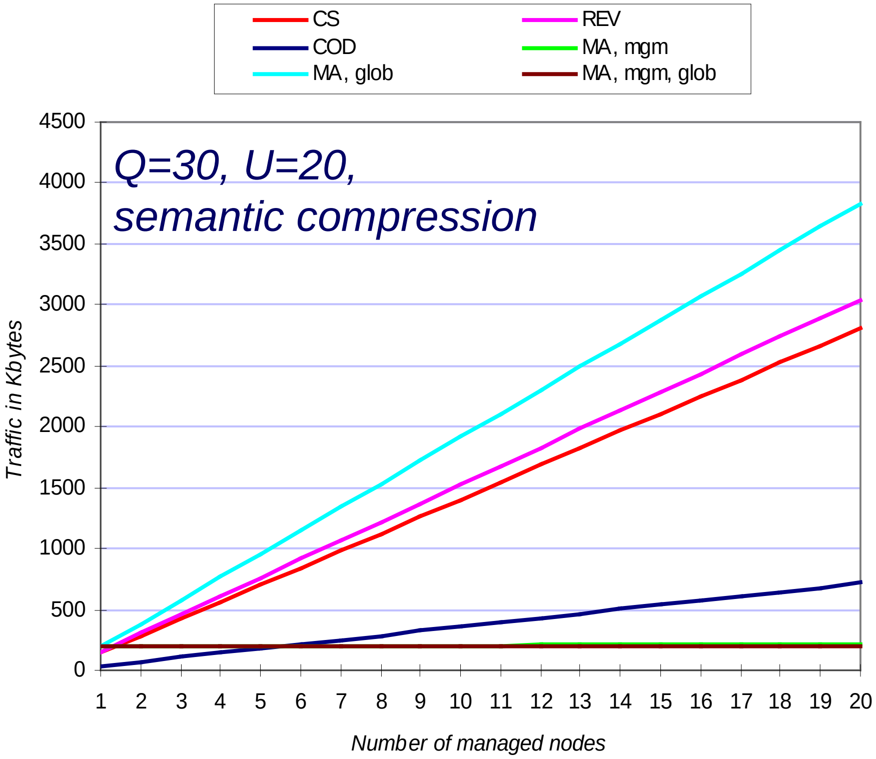
20 | 20 | up to 1460

$$\eta(X)X = 200 + \left[2\cdot 40\left\lceil \frac{(H_{ATP}+X)}{1460}\right\rceil + (H_{ATP}+X)\right] + (2\cdot 40 + A_{ATP})$$

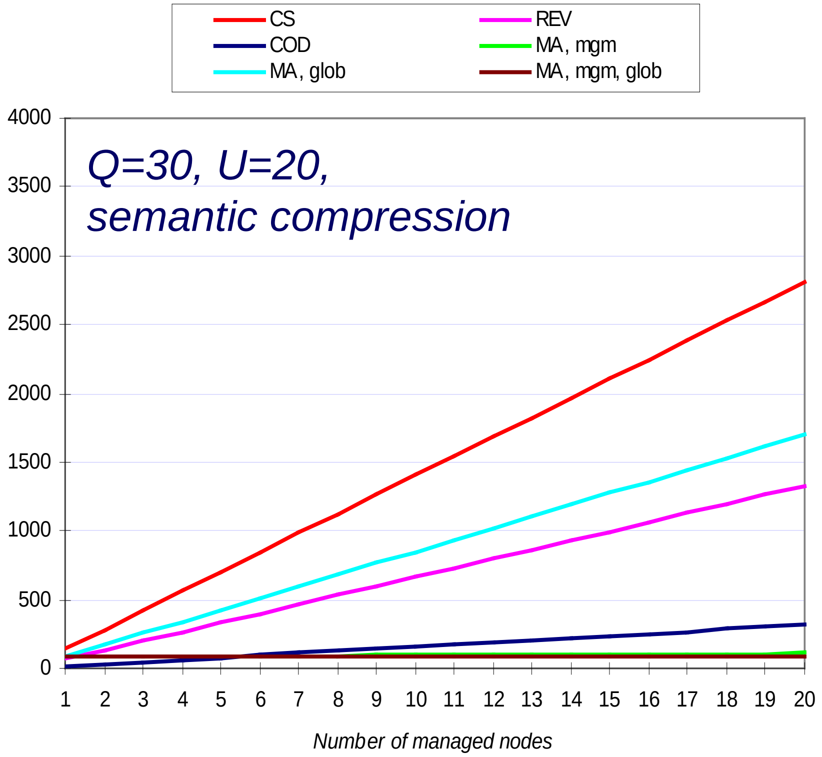
Politecnico
di Milano

Analyzing the Impact of Technology

SNMP vs. ATP



SNMP vs. TCP



Measured values: I=48 bytes, R=66 bytes
Measured code size: C_{REV}=5.6 Kbytes, C_{COD}=5.1 Kbytes, C_{MA}=6.6 Kbytes

Lessons Learned

- The analysis was relatively simple, as other quantitative and qualitative aspects were not considered
- Still, it provided deep insights into the nature of mobile code applications, by identifying:
  - the limits of applicability for the mobile agent paradigm
  - the impact of mechanisms and protocols provided by mobile code technology
  - precise criteria to choose the architecture optimizing a given cost function
  - solid background for argumenting in favor (or against) the mobile code option
- Moreover, it demonstrated also how this analysis is not only feasible, but also useful in a real world context

Lessons to Apply

- Scalability and fault tolerance are highly amenable to quantification
- Analysis like the one shown can help identify the tradeoffs between scalability, fault tolerance, complexity of implementation, and functionality
    - ➤ At the technological level, they can drive the decisions involved in the design of mobile code mechanisms
    - ➤ At the application level, they can help characterize the notion of scalability and fault tolerance in a given application domain, and there to evaluate the benefits brought by mobile code

Some (Bitter) Considerations

- Many systems embody design criteria that conflict with or hampers the exploitation of mobile code, e.g.:
  - Heavy application-level network protocols
  - Centralized code servers
  - Assumption of permanent connection
  - Migration policy hard-wired in the mechanism
- ***Key open issues:***
  - lack of validation of abstractions and mechanisms against real world applications
  - lack of quantitative characterization and comparison of mobile code technologies
- ***The above issues are hampering acceptance and credibility of mobile code research as a whole***

Conclusions

- Distributed applications can exploit mobile code to improve scalability and fault tolerance
- Mobile code systems must be designed to be scalable and fault tolerant themselves
- The notion of scalability and fault tolerance is not uniquely defined: it depends on the application domain
- Quantitative evaluations can be used to characterize a domain and to provide extremely useful insights for design criteria at both levels