

# Configurable Load Measurement in Heterogeneous Workstation Clusters

Christian Röder, Thomas Ludwig, Arndt Bode

LRR-TUM

Lehrstuhl für Rechnertechnik und Rechnerorganisation  
Technische Universität München, Munich, Germany

email: {roeder|ludwig|bode}@in.tum.de

## Agenda

- Motivation  
(Starting Point – Goal)
- NSR - The Node Status Reporter  
(Reference Model – Node Description – Design Goals and Aspects)
- Implementation Aspects  
(Client-Server Architecture – Library Calls – NSR Daemon – Communication Structure)
- Evaluation  
(Measurement Overhead – Embedding NSR into Load Management)
- Summary and Future Work

## Motivation

### Starting Point

- **experience:** on-line tools for message-passing based parallel applications
  - **examples:** system-integrated load management, visualization, etc.
  - **needed:** observation and manipulation of a distributed computing system
  - **target system:** COW, heterogeneous execution nodes, time-sharing usage model
  - **typical:** “load monitor process”, standard UNIX tools (`top`, `sysinfo`)
- ⇒ **scenario:**  $n$  parallel applications execute  
on  $m$  architecturally heterogeneous nodes  
controlled by  $c$  on-line tools

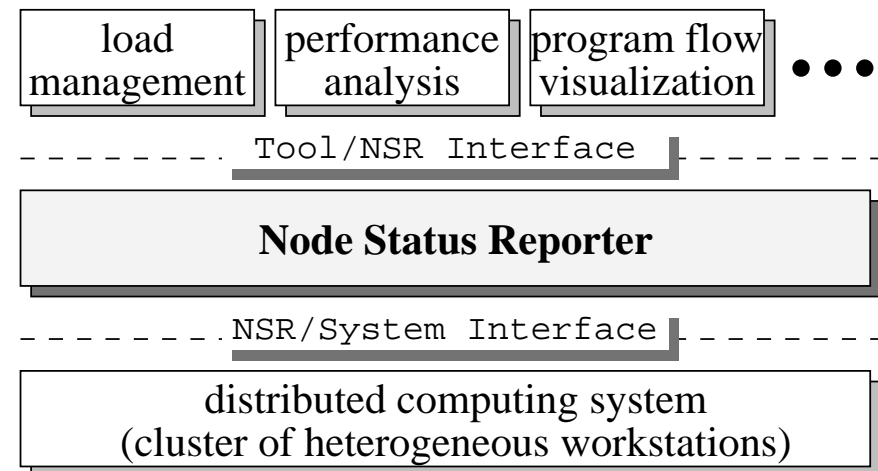
## Goal

- **dimension:**  $c$  on-line tools for  $n$  parallel applications often need  $m$  identical/different/overlapping sets of values describing system state
- **problem:**  $c \times m$  measurements result in severe overhead
- **approach:** detect and serve multiple value request,  
 $\Rightarrow$  reduce number of measurements to lowest possible limit
- **requirement:**
  - uniform measurement interface
  - efficient and flexible measurements
  - uniform data representation across different nodes
- **result:** independent monitoring component  
 $\Rightarrow$  reduce implementation effort to  $c + m$

# NSR - The Node Status Reporter

## Reference Model

- combines features of UMA and OMIS
- flexible data measurement component
- static and dynamic node description
- e.g. for load management:  
basis for adaptive load modeling



- **Tool/NSR Interface:**
  - communication between tools/applications and NSR (uniform interface)
  - tools/applications control the behaviour of NSR
- **NSR/System Interface:** data measurement, data scaling, data distribution

## Node Description

- exemplary overview:

type	static	dynamic
OpSys	name, version, node	ctxt, exec, sysc
CPU	arch, ncpus, bench	rql, avenrun, waitl
MEM	npages, pgsz, bench	usedpages, swapstats
I/O	disksize, swapsize	raw, nfs
NET	ipaddr, type, bandwidth	local/remote send/receive
users	maxusers	local/remote logins

- OMIS 2.0: `node_get_info()` (<http://www.bode.informatik.tu-muenchen.de/~omis>)

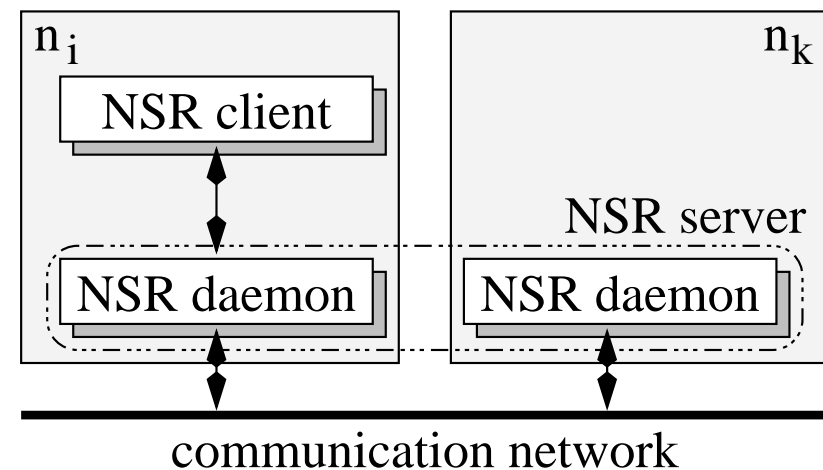
## Design Goals and Aspects

- **portability:** identical interface and adaptable to various architectures
- **uniformity:** measured values unified with respect to their units of measure
- **scalability:** increasing number of nodes must not slow down supported tools  
⇒ low measurement/distribution overhead, overlapping operations
- **flexibility:** Which data is needed? (static/dynamic node description)  
When is the data needed? (periodic/immediate measurements)  
From which set of nodes? (dynamic configuration)  
⇒ management and measurement control by applications
- **efficiency:** serve several tools/applications simultaneously  
⇒ synchronization and asynchronous operations
- **security:** selective user access restrictions to nodes

## Implementation Aspects

### Client-Server Architecture

- **NSR client:** process that initiates NSR library calls  
 ~> measurement configuration and data access
- **NSR server:** set of replicated NSR daemon processes  
 ~> cooperative request handling, data measurement, data distribution, data collection
- client communicates with **local** daemon  
 ~> "NSR master daemon"
- NSR master daemon distributes request  
 ~> "NSR slave daemon"
- "master" and "slave" **only** attributes



## NSR Library Calls

- **NSR Management:**

- `nsr_attach()`: attaches the caller to a set of NSR daemons
- `nsr_config()`: configures measurement (auxiliaries: set and test)
- `nsr_detach()`: detaches the caller from a set of NSR daemons
- variants: blocking or non-blocking calls; `nsr_exchange()`

- **Data Access:**

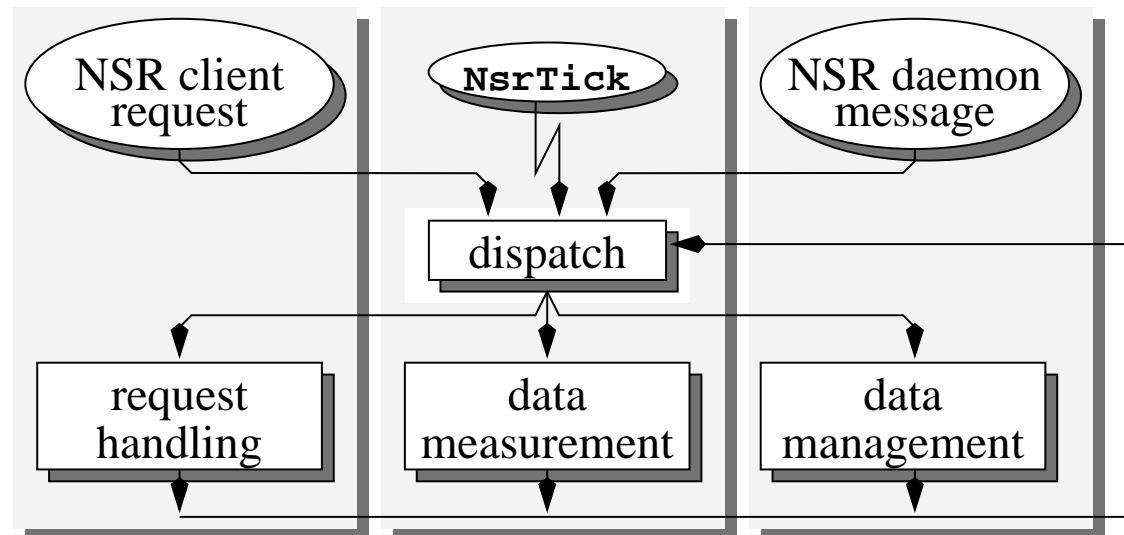
- `nsr_get_nodeinfo()`: immediately measure exactly once (blocking)
- `nsr_update_nodeinfo()`: periodically update measured values

- **Data Structure:**

```
typedef struct {  
    long cnt, tick;  
    static_hostinfo stthi;  
    dynamic_hostinfo dynhi;  
} nsr_info;
```

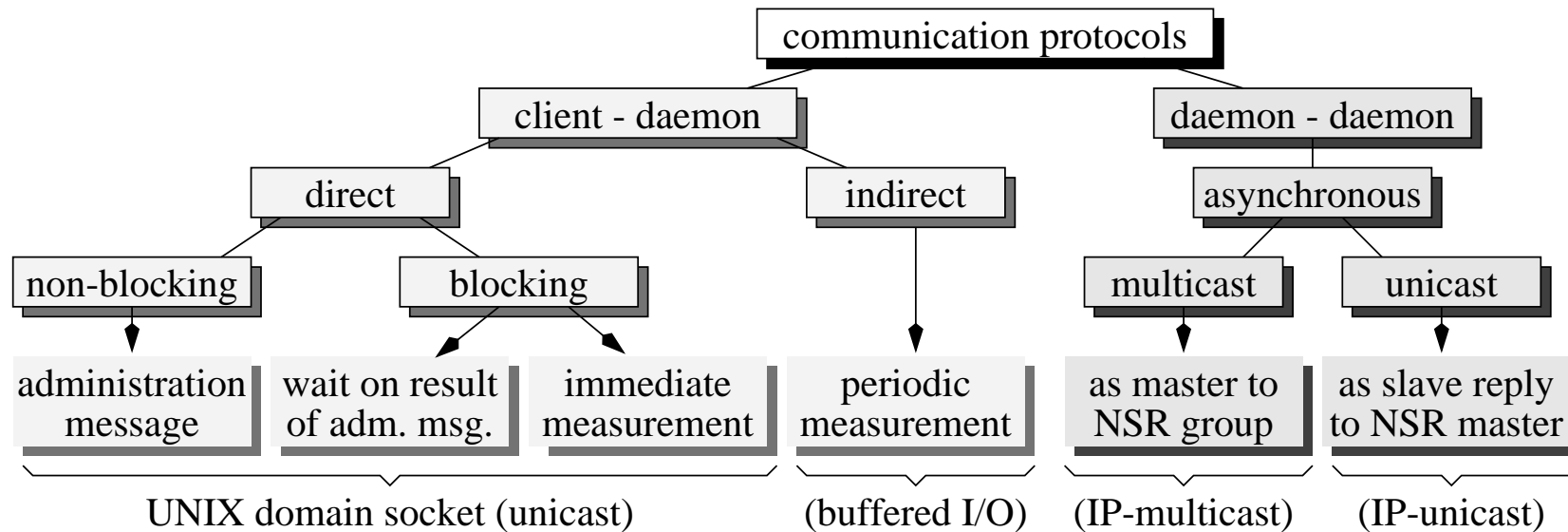
# NSR Daemon Functionality

... after an initialization phase (read static node description) ...



## Communication Infrastructure

### Implemented Protocols

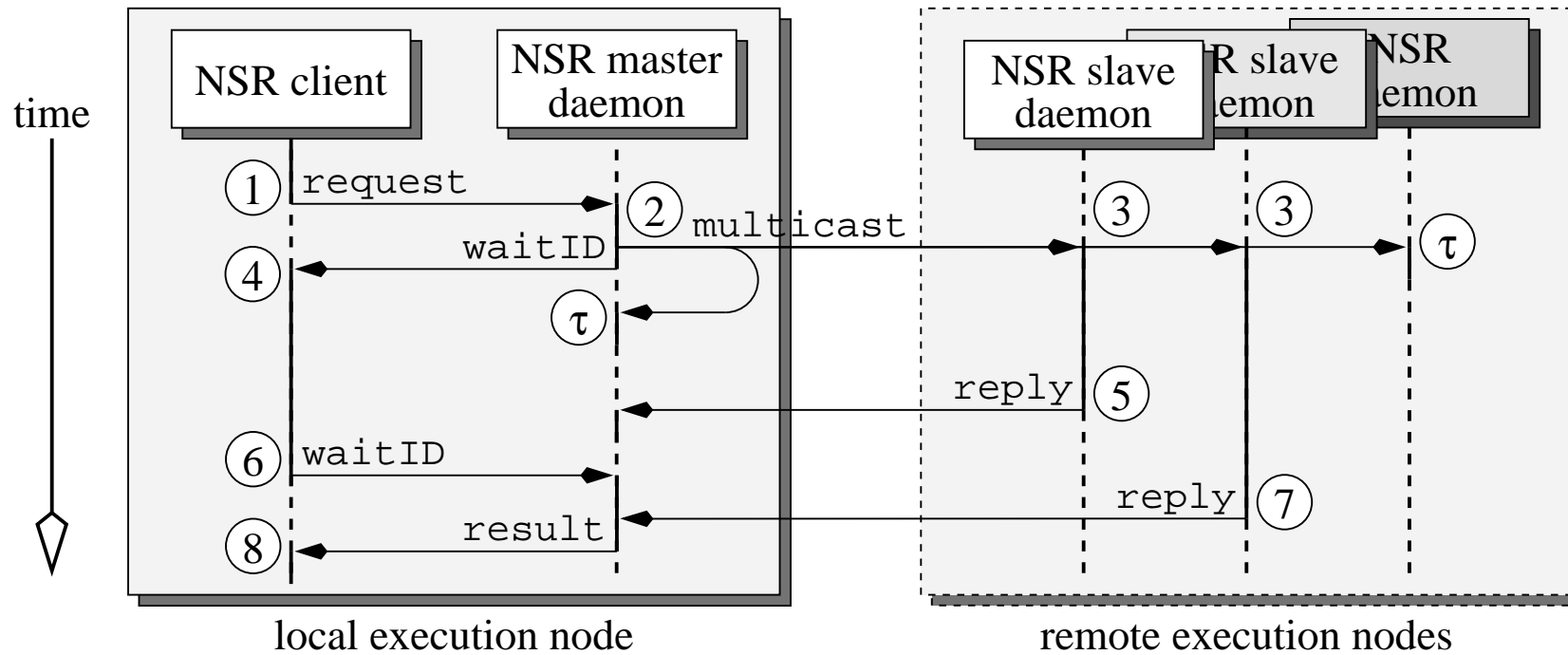


### IP-multicast:

- NSR server: NSR daemon processes constitute an IP multicast group
- one step towards metacomputing (not limited to a LAN environment)

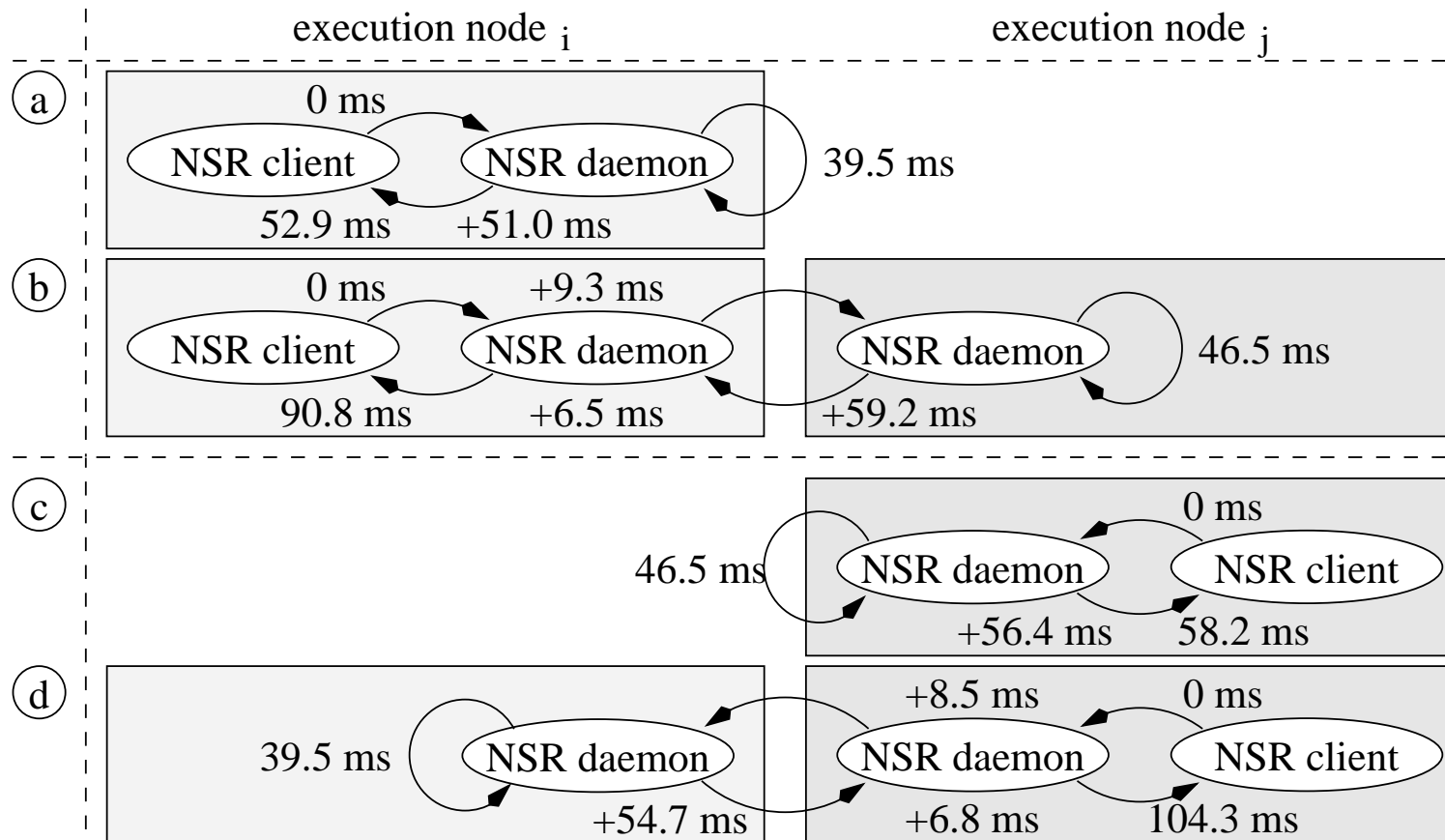
# Course of Asynchronous Operation

... a two phase protocol ...



# Evaluation

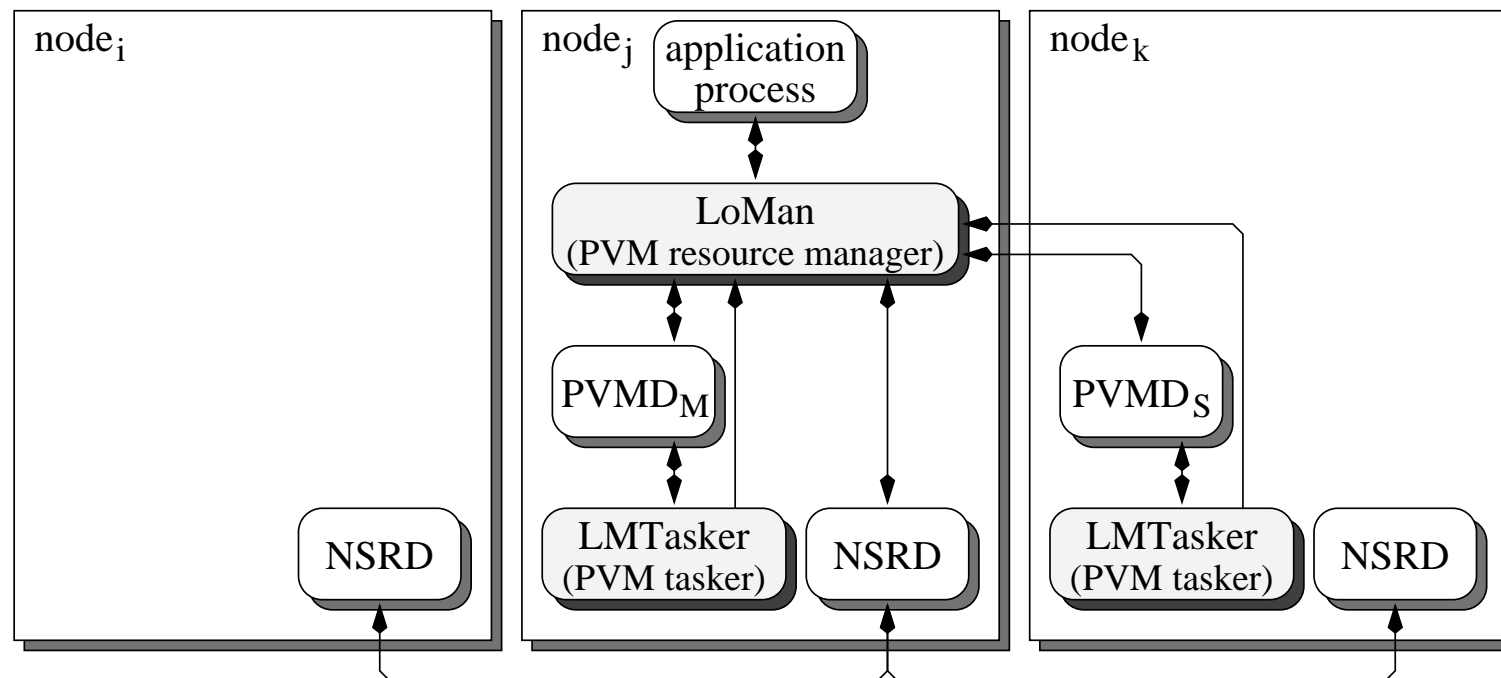
## Measurement Overhead



- performance of plain measurement routines comparable with public domain tools

## Embedding NSR into Load Management

**LoMan:** system-integrated load management for parallel applications that execute in heterogeneous workstation clusters



## Summary and Future Work

- NSR serves as a flexible data measurement component for diverse on-line tools, but is independent of any tool intend (does not interpret data)
- ready-to-run for SUN Solaris 2.5, Solaris 2.5.1, HP-UX 10.20, LINUX 2.0
- flexibility:
  - one NSR client can control several subsets of NSR daemons
  - NSR server can handle different NSR clients (from different users)
- future work:
  - porting the measurement routines to IRIX, AIX, Windows NT, etc.
  - GUI
  - available under GPL in about three weeks

<http://wwwbode.informatik.tu-muenchen.de/~nsr>

## Comments

- (2, 1) The need of a Flexible Status Measurement Component originates in a long term experience on tools for parallel systems at our char. Recently, skipping the focus from dedicated parallel machines (homogeneous, exclusive use of processing elements) to heterogeneous workstation clusters.
- (2, 2) standard UNIX tools have limited functionality, no data distribution
- (2, 3) what does this scenario mean for implementing appropriate monitoring components?
- (3, 4) By answering this question we can define the goals for this project
- (4, 5) here just a first impression, details will be shown on the next slides
- (4, 6) not only on-line tools, any kind of application (e.g. application integrated LM) can benefit from NSR
- (4, 7) it is important that NSR does not interpret the data, because it does not know the tools intention
- (5, 8) tools are interested in two types of information (static/dynamic) from all resource types, see table
- (5, 9) the efforts for developing the NSR directly influenced the specification of OMIS 2.0
- (5, 10) information about local/remote users necessary in time-shared systems
- (5, 11) benchmark values at startup time of nodes (kernel routine from the NAS parallel benchmark EP
- (7, 12) NSR server with dynamical changing number of NSR daemon processes, failure and recovery
- (7, 13) NSR client stateless, NSR server keeps all information about requests
- (8, 14) periodically update measured values from one node or a set of nodes
- (9, 15) the concept of alive and breakdown messages
- (10, 16) daemon-daemon communication uses XDR for different architectures, support for heterogeneity
- (11, 17) every multicast group member receives the message;  $\tau$  means received, but nothing to do
- (12, 18) average of 100 calls, note the difference between different speed of nodes