Middleware Support for Voting and Data Fusion

Dave Bakken
Zhiyuan "Troy" Zhan

Chris Jones
David Karr

School of Electrical Engineering
and Computer Science
Washington State University
Pullman, Wash., USA

Distributed Systems Department
BBN Technologies
Cambridge, Mass., USA

www.eecs.wsu.edu
www.dist-systems.bbn.com

© 2001 Dave Bakken: Voting Virtual Machine
Motivation

- The increasing demand of availability of online services can be met with replication of servers
- Common strategy: active replication
  - Request is sent to all of the server replicas and
  - Each replica will process the request and send reply
  - Must select 1 for client application to receive
- This selection process is called voting
- Collation and Data Fusion are more general than voting
  - No replicated servers
  - Values not necessarily supposed to be identical
  - Seems to be a fundamental pattern in distributed systems
Voting Definitions:

- **collation voting**: choosing (or creating) one reply (or request) from many
  - Differs from **synchronization voting**, a form of pessimistic concurrency control, also called “voting” in the literature

- **ballot**: one request or reply from one object replica (reply[1])

- **vote**: the process of choosing/creating one ballot from many

- **byte-by-byte voting**: voting algorithms compare marshaled parameter buffers on a byte-by-byte basis, unaware of data types, alignment, etc.
  - More on this later…
What is Needed

• Middleware support for voting
  – Allow voting on any parameter or combination of them
  – Separate voting language for reuse, analyzability
  – Hooks to allow changing of voting algorithms at runtime
  – Voting management to provide client transparency
  – Not byte-by-byte voting

• Applies to more than voting: collation & data fusion
  – Intrusion detection: multiple different probes monitoring a host/LAN/domain
  – Ad hoc mobile network protocols
  – Distributed sensor networks
  – Parallel neural nets to estimate power grid margins
  – Hierarchical resource monitoring and aggregation
  – Merging object state from caches or partitioned replicas
Outline of Presentation

• Overview of voting, collation, and data fusion
• **Current middleware voting: limitations & their architectural implications**
• Voting Virtual Machine (VVM) architecture
• Related work & Conclusions
Limitations of Byte-by-Byte Voting

- CORBA provides interoperability across
  1. CPU architecture
  2. Operating system
  3. Programming language
  4. ORB (vendor) implementation

- CORBA’s “wire protocol” CDR:
  - Uses IEEE 754 encoding for floating point values’ transmission
  - But with different architectures (and other 3 variables same) can have different internal precisions and roundoff differences …

- Some Pentium models have HW rounding switch
  - Thus all 4 above can be same, but still values diverge!
Byte-by-Byte Limitations (cont.)

• Have to look at the IDL to be able to handle
  – Variable-length header information which an ORB implementation may fill in or leave out
  – Otherwise alignment off
  – Example: system context
  – CORBA spec states value of padding bytes is undefined, anyway!

• Byte-by-byte voting in practice:
  – Developers have no problems in lab or single LAN, because very homogenous (CPU, OS, language, ORB)
  – When fielded or released, mysterious bugs start to show up because much more heterogeneous
  – This is true for all middleware, not just CORBA!
Byzantine Architectural Implications

**Ideal:**
- Middleware
  - TCP
  - Secure
  - Reliable
  - Multicast
- IP

**Reality:**
- Middleware
  - Secure
  - Reliable
  - Multicast
  - marshal
- IP

→ No clean layering: transport needs help unpacking data for value check!

- Additional result: byzantine reliable multicasts
  - Cannot do value checking without help! (if heterogeneity)
    - (except from single `int` perhaps)
    - Applies to Rampart, “Practical” (MIT), …
Outline of Presentation

• Overview of voting, collation, and data fusion
• Current middleware voting: limitations & their architectural implications
• **Voting Virtual Machine architecture**
  – Voting Virtual Machine (VVM)
  – Voting Definition Language (VDL)
  – VDL Example
• Related work & Conclusions
Voting Virtual Machine (VVM)

Novel features:

• VVM performs voting on application-level data
  – First to not use byte-by-byte voting that we are aware of
  – Embeddable into CORBA, DCOM, other kinds of middleware (MOM, publish-and-subscribe, XML-RPC, ...)
  – Voting module thus **not** embedded in the application
    • Could be used by a application directly if desired

• Voting definition Language (VDL) allows the coding of portable, reusable, stand-alone voting algorithms
  – Supports *dynamic voting* (accounts for membership changes during a vote)
  – First voting or data fusion language for middleware we are aware of
VVM Architecture

\[
\text{iiop\_msg}[1..N] \quad \xrightarrow{\text{unmarshal}} \quad \text{voter core} \quad \xrightarrow{\text{voted\_param}_k} \quad \text{marshal} \quad \xrightarrow{\text{voted\_iiop\_msg}}
\]

- Event: Failure notification
- Current voting policy & params: weights
- Voting statistics
- Current conditions & QoS requirements
- Alerts to subscribers

Voter Manager

Voting Status Service (VSS)
VVM Failure Model

- Client and server hosts: crash failures
- Server applications: Byzantine
- Server systems software: crash
  - Could secure with network attachment controller a la Delta-4 NAC
  - Could use stronger Byzantine protocols a la Rampart or "Practical" from MIT (being done now with TIS...)
Voter goes through 3 states:
- **quorum**: wait to vote
- **exclusion**: toss out some ballots
- **collation**: choose one of the remaining

Branching Exceptions Confidence Values
VDL Primitives (cont.)

• VDL primitives for quorum, exclusion, collation
  – As described, see paper
• Exceptions & branching: conditions based on
  – Elapsed time since first ballot
  – Number excluded
  – Percent excluded
  – Number remaining after exclusion
  – Standard deviation after exclusion
  – …
• Confidence values
  – Above conditions for exceptions
  – More TBD …work in progress….
VDL Syntax Overview

policy name [ (parameters) ] {

quorum (quorum_op (params))
   [throw ex_name if (condition) ]*

[exclusion [label] (exclusion_op (params))
   [replace by mean|median|(value)]

   [ [throw ex_name if (condition) ]
   goto quorum
   [(using (quorum_op (params)) more|total)]
   if (condition) ]
   ]*
}*

...
VDL Syntax Overview (cont.)

...  

\[
\begin{align*}
\text{collation} & \quad (\text{collation}_\text{op} (\text{params})) \\
& \quad [\text{throw ex name when (condition)}] \\
& \quad [(\text{using quorum}_\text{op} (\text{params})) \text{ more|total}] \\
& \quad [\text{goto quorum label}] \\
& \quad [\text{goto exclusion label}] \\
& \quad [\text{(using exception}_\text{op} (\text{params}))]] \text{ if (condition)} \\
& \quad [(\text{confidence}_\text{expression})] \\
\end{align*}
\]
Confidence Values

- Problem: returning a vote or an exception are two extreme choices!
- Issue: how “good” was the vote?
- Confidence values used in neural networks and fuzzy logic
- Idea for VVM:
  - Return another value(s) “on the side”, like Unix errno
  - Optionally used by client or perhaps a QuO delegate or other intermediate layer to adapt...
Supermajority Example

• Problem Statement
  – Return a ballot if 60% of the ballots are within 1% of the median of the ballots received so far
  – Only wait 5 seconds
  – Ballot returned should be median from those within 1%

• Note: Items underlined can be parameters to policy
  – Not shown here for simplicity
SuperMajority VDL Code

policy SuperMajority {

  quorum (until (60 %))
    throw QUORUM_TIMEOUT if (elapsed_time > 5000)

  exclusion (distance (1 %))
    throw BAD_VALUES if (pct_excluded_total > 40)
    goto quorum (using (until (1)) more)
      if (pct_remaining_total < 60)

  collation (median)

  confidence (ballots_remaining / ballots_received)
}


Outline of Presentation

• Overview of voting, collation, and data fusion
• Current middleware: limitations & their architectural implications
• Voting Virtual Machine (VVM) architecture
• Related work & Conclusions
Related Work

• Way too much to list here or even in (this) paper…. 

• Synchronization Voting
  – First by Thomas of BBN (1979)
  – Weighted Voting (Gifford, 1979)
  – Generalizations: multidimensional, etc.
  – Mostly (except NMR) for databases

• Consensus service (EPFL)
  – Different architecture: voting client not one providing a value
  – voted_reply might not be one of input ballots (e.g., mean)
Related Work (cont.): Collation Voting

- SIFT
- NMR Systems
- N-Version programming
  - No votes on requests or individual reply parameters
  - Very limited quorums
- Recent CORBA replication: Electra, Eternal (UCSB, not FT-CORBA version), AQuA
  - No votes on requests or individual reply parameters
  - Very limited quorums
- Immune
  - Does allow voting on client requests and reply parameters as a block

• All above use **byte-by-byte** voting & no VDL
VVM Collaborations and Tech. Transfer

- Analysis and Management: Doug Blough et al. Georgia Tech. (Fast Abstract later today)
- Network Associates Inc. Labs (a.k.a. Trusted Information Systems)
  - Integrating VVM subset into TAO for byzantine-fault-tolerant CORBA
  - Replicated clients so replicated voting
  - Project ITDOS on DARPA OASIS program (Jay Lala)
- TriGeo Network Security Inc. adding to commercial product for managed security and intrusion detection
- BBN Technologies APOD project in Oasis program evaluating VVM
Future Work

• Extend for intrusion detection usage
• Small device profiles and distributed sensor networks
• Refine VDL to be more general, consistent, …
  – Multidimensional: median $\rightarrow$ centroids, distance $\rightarrow$ distance function, etc.
  – Generality: more statistical operations
  – Application-transparent multi-parameter voting
  – Orthogonality: vote on arrival times, too?
  – Expressive power: formal semantics
  – Safety: guarantee all VDL policies terminate
    • Remove or restrict gotos?
    • Use recursive policies instead?
VVM and Fault Tolerant CORBA

- FT-CORBA not required to tolerate value faults
  - Placeholder replication style ACTIVE_WITH_VOTING defined but not yet specified
  - Spec could allow a customizable voter module, but unlikely

- All replicas of an object must be hosted by infrastructure from same ORB vendor
  - Could zero out padding bytes and rely on this
  - Thus byte-by-byte voting could work on integer types
  - Would still have to deal with floating point (inexact voting)

- VVM + FT-CORBA???
  - “You’re dead if you standardize beyond the state of the practice” Rick Schantz, BBN
  - VVM and voting in middleware is recent state of the art
  - FT-CORBA would want a small VVM subset, maybe no VDL
Conclusions

• VVM can be embedded into any kind of middleware substrate we are aware of
  – Just port the marshal and unmarshal modules
• VVM performs voting on actual application-level data parameters, not marshaled parameter buffer in a byte-by-byte fashion
• VDL allows specification of voting algorithms
  – To control the VVM
  – Portable & reusable
• URLs
  – vvm.eecs.wsu.edu
  – www.eecs.wsu.edu/~bakken/middleware.pdf
VDL Primitives

- quorum
  - `until k [%]`
  - `all_but k`
  - `random n m [%]`

- exclusion
  - `lowest n [%]`
  - `highest n [%]`
  - `furthest n [%]`
  - `distance e [%]`
  - `outside_sigma x`
  - `distance_neighbor d`
  - `distance_cluster d`
  - `cluster_support d p [c]`
  - `inner k`
  - `nearest k`
  - `random n [%]`
  - `none`
VDL Primitives (cont.)

• collation
  – median
  – mean
  – mean_neighbor
  – mode
  – random
Weighted Voting in VDL

• Weighted voting allows non-equal treatment of ballots from different replicas
  – Main use: security in voting on (supposedly) identical replicas
  – Also can also add randomness to collation and data fusion

• Operations currently in 2 voter core states:
  – quorum: wait for points, not ballots, where each ballot’s arrival is \(\geq 1\) point
  – collation: expand remaining ballots based on a weighting, then do mode, median, ...
  – collation example: if ballots contain \{2,3,4\} and weights are \{1,2,2\}, then expand to \{2,3,3,4,4\} and then do median or mode or mean or random or ...
VVM Development Environment

- CORBA Interface Repository (IR)
  - iiop_msg[1..N]
  - lookup
  - paramk[1..N], param_types
  - lookup
  - voter core
    - voted_param_k, param_types
    - marshal
      - voted_iio_msg

- VDL files
  - vdl2vvm
  - vdl2analysis
  - Vote Manager
    - current voting policy + params

- server IDL files
  - idl2vvm
  - ir2vvm
  - param_types lookup table