Hierarchical Adaptive QoS Control for Voting-based Data Collection in Hostile Scenarios

Mohammad Rabby & Kaliappa Ravindran
Graduate School and University Center (CUNY)
New York, NY 10019
Email: ravi@cs.ccny.cuny.edu; mfrabby@yahoo.com

Kevin A. Kwiat
Air Force Research Laboratory
Rome, NY 13441, USA
Email: kwiatk@rl.af.mil

***Abstract*—Voting among replicated data collection devices is a means to achieve dependable data delivery to the end-user in a hostile environment. Here, a QoS prescription is about how often a correct data is delivered to the user in a timely manner. The paper deals with situations where the available network bandwidth varies dynamically and/or the environment fault parameters change unpredictably. A *hierarchical adaptation* method is employed to manage the QoS at two levels: the core voting module and a situational assessment module. The hierarchical QoS control enables an optimal use of resources by the voting system vis-a-vis the mission-critical application needs.**

I. INTRODUCTION

Majority voting among replicated sensor devices is a mechanism to decide on the delivery of a correct data to the end-user, in the presence of device failures and attacks [1]. An application scenario is the sampling of remote terrain data at regular intervals for surveillance purposes. Replica voting is typically realized by a 2-phase protocol: the proposal of candidate data by a device (first phase), and the collation of votes from the remaining devices about their consent/dissent on the data proposed (second phase) [2]. Here, a device decides on its consent/dissent based on how its locally computed data from the environment matches with the candidate data proposed. The voting is coordinated by a centralized secure entity, which decides about the final delivery of a data to the end-user based on the consent votes received. See Figure 1.

The data being voted upon can exhibit large dimensionality, and take long time to be generated (e.g., terrain surveillance images). Furthermore, the network may exhibit a wide range of loss/delay behaviors, with the available bandwidth for data transport varying dynamically — as in airborne networks shared with many other applications. The behavior of faulty devices may itself be random, ranging from benign levels to malicious acts. In the absence of complete knowledge about the environment, the voting is often geared to deal with certain worst-case situations by deploying a conservatively large number of replicas (say, 8-10). Our paper is on the *external management* of such a replica voting system from QoS standpoint, where the voting protocol and its underlying system-level infrastructure is treated as a black-box with concrete I/O interface to facilitate the signaling and control of

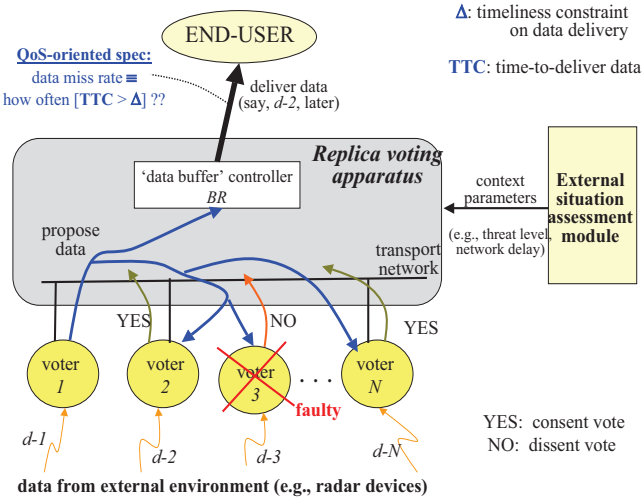

Fig. 1. QoS-oriented view of a replica voting system

voting operations. This enables instantiating the voting black-box with parameters for use in specific application settings.

Our earlier works on a voting system have engineered the protocol and system mechanisms to realize a desired quality of data delivery (QoS), prescribed in terms of a timeliness constraint [3]. This involved studying the relationship between various operational parameters of the voting system (such as timeout values, number of replicas, network/system bandwidth, and broadcast window sizes), the computational and communication asynchrony parameters (such as message delays/loss and computational delays), and the system performance indices: time-to-complete voting (TTC) and data/control message overhead (MSG).

A situation-aware management of replica voting involves determining the protocol and infrastructure parameters to meet the QoS adaptation needs under the prevailing environment conditions. Such a voting system is viewed as made up of:

1) A core voting system that provides the basic QoS adaptation during delivery of sensed data to end-users;
2) A situation-aware manager that parameterizes the voting module as per the environment conditions.

In this hierarchical structure, the sub-system 2 provides a context for the internal operations of sub-system 1. For instance,

978-3-901882-48-7

the number of replicas N is treated as a fixed parameter at the voting protocol level. However, at the situational management level above, N is treated as a control variable (as determined by the prevailing threat level). This is because device-level replication incurs administrative costs, and often imposes higher demands on the system resources — thereby affecting the QoS of other applications sharing the resources. These costs are impacted by situational parameters: such as threat levels, network conditions, and application priorities. The multi-level structuring of adaptations allows an analysis of the composite behavior of voting system in a *holistic manner*, i.e., in totality of the context provided by the situational environment, application needs, and voting protocol core.

The paper is organized as follows. Section II discusses our prior works and other related works. Section III outlines the voting system core mechanisms from a QoS standpoint. Section IV describes a framework for situation-based management of the voting service. Section V concludes the paper.

II. Our prior works and related works

Our recent works examined how the system parameters can be adjusted at run-time to deal with the dynamically changing external conditions: say, a large increase in message loss due to an elevated level of attacks on the network [4]. Depending on the severity of external conditions, the dynamic *adaptation* mechanisms range from simple parametric adjustments (e.g., changing the timeout values) to complex operational changes in the protocol modes (e.g., switching from a light-weight voting protocol to a robust one) [5], [6]. These works studied the protocol-level mechanisms to infuse dynamic adaptation functionality in a voting system: namely, middleware interface and end-to-end QoS monitoring. In contrast, our present paper deals with the external management of adaptation.

Many existing works have studied QoS adaptation in data/content networks: some focusing on *service-oriented* specification & control at the application layer (as in [7], [8]), and others on integrated domain-specific mechanisms for the application on hand (such as radar tracking QoS, as in [9], [10]). In this light, the 2-phase commit based replica voting, as studied in our paper, is more of an application-level distributed protocol with fault-tolerance goals [2], [11] — rather than a data/content networking mechanism *per se*. Nevertheless, the infusion of replica voting in contemporary network settings with resource constraints (such as wireless sensor networks and airborne data-relay networks) reaps benefits from the existing methods for service-level QoS adaptation.

III. QoS-oriented view of replica voting

In this section, we discuss how a QoS-orientation can be cast into the voting protocol functional elements and how the required adaptation mechanisms are incorporated therein.

A. Controllable parameters of voting system

The data delivered to the user as representing an external event may be associated with a timeliness parameter Δ. It pertains to how soon the data produced by a sensor device be delivered at the user since the occurrence of event represented by this data (i.e., life-time of data). The data generated by a device may also deviate somewhat in content, relative to the actual reference datum, i.e., the ground truth. This is due to the inherent sampling errors in the sensing mechanism and/or resource limitations on the data pre-processing algorithms of the device (such as CPU cycles and memory sizes). A data is said to be accurate if its content does not deviate from the ground truth beyond a tolerance limit ϵ.

The data collection devices are replicated, to counter the effects of content corruptions and timeliness violations that may occur in the untrusted external parts of the system. The system employs a majority voting on the data fielded by various replicas [2] to decide on the delivery of a correct data to the end-user in a timely manner. With N replica devices ($N \geq 3$), we have $1 \leq f_m < \lceil \frac{N}{2} \rceil$, where f_m is the maximum number of devices that the voting protocol assumes as being faulty. The behavior of a faulty device v' may itself be random, ranging from benign levels to malicious acts. The fault severity of v' is captured by a parameter r: the probability that v' exhibits a faulty behavior during protocol execution ($0.0 < r \leq 1.0$). The data delivery by a replica voting system is deemed as correct if both the Δ and ϵ constraints are met.

B. Decentralized operations of voting protocol

The voting employs a 2-phase mechanism: the proposal of candidate data by a device (first phase), and the collation of votes from the remaining devices by a centralized secure entity for decision-making (second phase). The voting, in its basic form, requires the sending of consent and dissent votes (YES and NO) by devices about a data being voted upon, to a central site BR. Refer to Figure 1. Suppose a data $X(v)$ proposed by voter v is put to vote. Thereupon, a voter v'' sends YES or NO message to BR based on whether its locally computed data $X(v'')$, if ready, matches with $X(v)$ or not. Based on the YES and NO messages received from $\{v''\}$, BR determines if $X(v)$ enjoys a majority consent for delivery to the user. The solicitation of votes from devices gets repeated until at least $(f_m + 1)$ consent votes are received and the time elapsed does not exceed a latency limit Δ.

The central site BR is assumed to be immune from attacks, providing a tamper-free data proposal/delivery handling and vote collation activities. The voter devices and BR are connected through a secure multicast message channel, where communications are authenticated, have certain minimum bandwidth guarantees, and enforce anonymity among voters. Message authentication, say, using MD-5 type of digital signatures [12], prevents a malicious device from stealing the identity of non-faulty devices, staging replay attacks, and the like. The data-oriented operations of a voter are deterministic: which means that a non-faulty device always produces a correct data (i.e., a data matching the ground truth) and always detects an incorrect data from another device by content comparison with its locally generated datum. Given the underlying assumption $f_m < \lceil \frac{N}{2} \rceil$, the voting protocol thus needs only to deal with software-level malicious activities

of v (if any) such as data corruptions, violation of protocol state-machine rules, and omission failures[1]. Here, an implicit assumption is that the actual number of failed devices f_a does not exceed f_m, i.e., $0 \leq f_a \leq f_m$.

C. Resource control in replica voting

In our earlier works [3], [6], we engineered variants of the voting protocol to achieve performance. Reduced message overhead (i.e., low bandwidth consumption) and low-latency data delivery are the performance goals, in the presence of uncontrolled behaviors in the data collection infrastructure: random message loss/delay, device-level faults, and computational asynchrony of devices. For instance, when $f_m \ll \frac{N}{2}$, $r \ll 1.0$ and $\sigma_{T_c} \approx 0$ (where T_c is the time for device-level computation of data), the decision on delivering a candidate data is made quickly and with less message overhead, because it is highly likely that the candidate data is non-faulty and a receipt of just $f_m + 1$ YES messages suffices for the decision (instead of $\lceil \frac{N}{2} \rceil$ YES messages). Such a performance-conscious voting protocol enforces a high quality of data delivery, namely, a large fraction γ of the data collected gets delivered, i.e., TTC $< \Delta$ with a probability γ (say, $\gamma > 0.9$).

γ depicts the extent to which the end-user's expectation on the voting system for a timely delivery of data is met (i.e., 'data availability'). From the standpoint of voting system, the data miss rate ζ, i.e., $(1 - \gamma)$ can be reduced by increasing N and B. This is because a resource allocation $[N, B]$ offers certain processing and message transport guarantees from the system infrastructure, to achieve predictable levels of TTC.

We assume a generalized notion of *partial synchrony* in the system: i.e., if an activity starts (say, a network message transmission), it will eventually complete in a finite amount of time. An upper bound on the completion time is however not known to the algorithm running on the system [13]. The synchrony property allows realizing the voting protocol without correctness violation. Furthermore, the extent of communication synchrony in message transport and computational synchrony among voter devices impacts the system resource usage: higher synchrony implies more resources.

We employ a decentralized 'listen-and-suppress' technique using randomized wait to reduce the likelihood of redundant proposals, in the presence of spontaneously generated data from voters. The random wait of a voter before proposing is based on the device-level asynchrony parameters $[\mu(T_c), \sigma(T_c)]$ and network message loss/delay $[l, \mu(t_{net})]$ [4].

D. Experimental results on 'replica voting' service

The performance parameter TTC (and hence γ) critically depends on the external environment parameters: $[f_m, r, l]$, the device-level computational asynchrony parameter $[\mu(T_c), \sigma(T_c)]$, and the network bandwidth B allocated for the voting operations. The graphs in Figure 2 shows how the QoS parameter $\zeta = (1-\gamma)$, i.e., data miss rate, varies with respect to f_m and l for the 2-phase voting protocol. The results are obtained from experiments carried out on a prototype implementation of the replica voting system on UNIX-based IP network platforms (see [6] for the protocol and system details) — which are corroborated by a probabilistic analysis. The exponential increase in ζ with an increase of f_m or l arises from a potentially large increase in the wasteful operations and/or message transmissions.

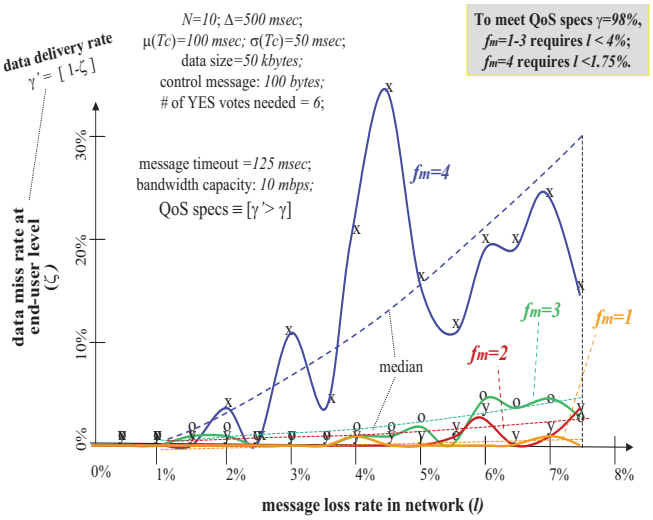

Fig. 2. Data delivery QoS relative to external environment $[f_m, l]$

The experimental results exemplify a black-box view of the voting system, with the I/O relationship captured through the graphs shown. Our interest is in the monotonic convex behavior of system performance with respect to $[l, f_m]$ — we had assumed $r = 1.0$ in the experiments. The I/O-style mapping relation can give a reasonable estimate of $[B, N]$ to achieve a desired TTC behavior.

A resource allocation $[N, B]$ has its own cost however — such as device installation & maintenance and QoS reduction for other applications (due to diversion of shared resources).

IV. External Management of Replica Voting

We treat fault-tolerance as just another QoS attribute that exhibits a continuously varying behavior (instead of a binary thing), which allows subjecting it to our extended notion of dependability. With this focus, we describe the infusion of adaptive fault-tolerance behavior during data collection from replicated sensor devices in a hostile environment. Figure 3 shows the system-level structure.

A. Management view of voting-based data collection

For management purposes, the voting system is represented as a black-box with a mapping function:

$$B = \mathcal{F}(\gamma, e_v, [N, f_m]), \quad (1)$$

[1]The failure containment within the software functions of a voter and the attack-immunity of BR preclude the need to consider byzantine failures. This ensures the correctness of voting protocol when $N > 2f_m$ (in contrast with the byzantine model of voting where $N > 3f_m$). The fault model assumed in our system is thus stronger than 'fail-stop' behaviors, while the system is structured in such a way to prevent byzantine failures.

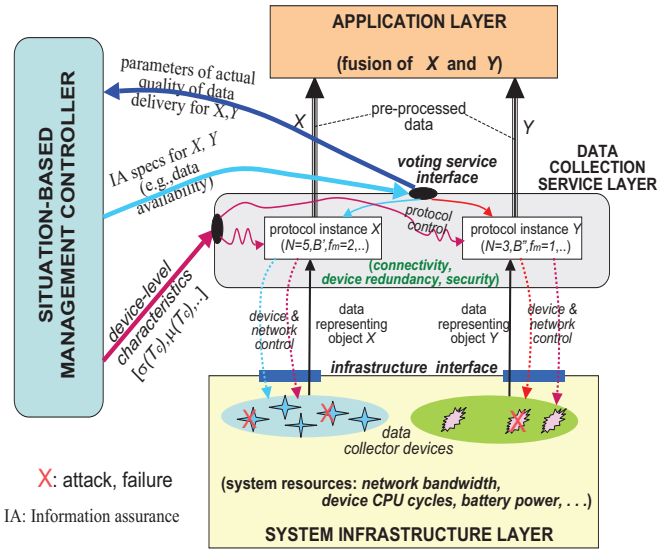

Fig. 3. Management structure of a fault-resilient data collection system

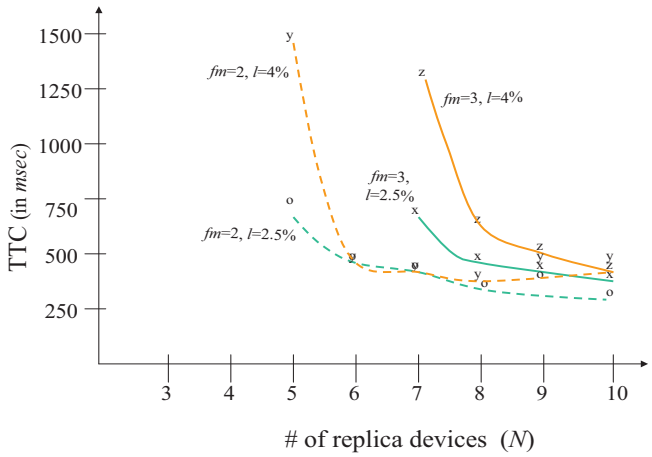

Fig. 4. Impact of device redundancy on TTC (experimental results)

where B is an estimate of the resource allocation needed in the voting system (namely, network bandwidth and device CPU cycles) to realize the QoS specs γ (i.e., the extent data miss due to violation of latency constraints) under the environment conditions $e_v \equiv [f_a, l]$ and device replication N. In this black-box view of voting system, we assume that $f_a \leq f_m$: i.e., the actual number of failed devices f_a does not exceed the upper limit f_m programmed into the voting protocol.

A closed-form representation of $\mathcal{F}$ (which is omitted for brevity) is often based on a probabilistic and/or operational analysis of the voting system. With a coarse representation of the system I/O mapping, it is likely that the actual resource needs B' to realize γ is not the same as the estimated needs B. Alternately, if B is the resource allocation in the voting system, the actual QoS achieved is:

$$\gamma' = \mathcal{F}^*(\gamma, B, e_v, [N, f_m]).$$

The management module determines the resources $[N, B]$ dynamically, over multiple 'observe-adapt' steps, to make $\gamma' \approx \gamma$. Possibly, $\gamma' < \gamma$ under severe resource constraints.

Procedurally, the adaptation functionality of voting system core is exercised by the situation-based management module with parameter inputs $[N, B]$. The initial values of $[N, B]$ are determined by the functional relation (1). Thereafter, the management module adjusts $[N, B]$, subject to resource availability, based on the observed data miss rate vis-a-vis the desired QoS γ. After one or more such observe-adapt cycles, the system reaches a steady-state with a sustainable QoS γ'. The situational assessment module itself gathers information about the external environment (such as threat level, network conditions, and application priorities) at slower time-scales.

B. Experimental results for voting protocol management

We study the influence of the device redundancy parameter N on the QoS achieved (while keeping the other parameters fixed). We reason that a higher N increases γ', *albeit*, in a non-linear fashion. Intuitively, over a region of small N (say, $N = 3 \cdots 5$), an increase of N has a strong impact on increasing γ', because the higher device-level asynchrony increases the likelihood of receiving $f_m + 1$ YES responses quicker (thereby lowering TTC). But, over a region of large N (say, $N = 9 \cdots 11$), the device-level asynchrony induced by an increase of N results only in a marginal reduction in TTC (assuming that f_m does not change with N). Our experimental results, as shown in Figure 4, corroborates this observation.

The variation of QoS γ' with respect to the situational parameters $[f_m, l]$ and protocol parameters $[N, B]$, as shown in Figure 2, depict a black-box view of the voting system. This I/O-mapping is representable as a tabular form in the situational assessment module for system management purposes.

V. Conclusions

Our paper employed hierarchical adaptation methods to manage the QoS at various levels of a replica voting system: namely, the timely delivery of correct data to the end-user. The fine-grained adaptations occurring in the voting system core are controlled macroscopically via the QoS reconfiguration actions occurring in application layer, in a context of the external events sensed by a situational assessment module.

Our past works had focused on engineering the core voting protocol mechanisms to lower the time-to-compete a voting (TTC) and the bandwidth consumed therein. In the current paper, we considered the dynamic nature of network bandwidth availability to support voting operations and the unpredictable changes in device-level fault occurrence and network message loss/delay. Here, the voting system is treated as a 'black-box' with known I/O behaviors, which is then exercised by the situational assessment module for a macroscopic control.

Our hierarchical adaptation approach offers the potential to keep the data miss rate of the voting system within the allowed limits as the system & environment parameters change. The paper provided experimental results on the voting system behavior to demonstrate the usefulness of our approach.

ACKNOWLEDGEMENT

The paper have been approved by AFRL for public release (distribution unlimited): 88ABW-2009-4460, on 21 Oct. 09.

REFERENCES

[1] H. Kopetz and P. Verissmo. **Real Time Dependability Concepts**. Chap.16, *Distributed Systems*, S. Mullender, Addison-Wesl. Co., 1993.

[2] P, Jalote and et al. **Atomic Actions on Decentralized data**. Chap. 6, *Fault-tolerant Systems*, John-Wiley Publ. Co., 1995.

[3] K. A. Kwiat, K. Ravindran, and P. Hurley. **Energy-efficient Replica Voting Mechanisms for Secure Real-time Embedded systems**. in proc. Intl. Conf. on *World of Wireless and Mobile Multimedia* (WoWMoM'05), IEEE, Taormina (Italy), June 2005.

[4] K. A. Kwiat, K. Ravindran, Jiang Wu, and A. Sabbir. **Adaptive QoS Mechanisms for Dependable Data Delivery in Replication-based Voting Systems**. In proc. SCS-SPECTS'06, Calgary (Canada), June 2006.

[5] K. Ravindran, K. A. Kwiat, A. Sabbir, and B. Cao. **Replica Voting: a Distributed Middleware Service for Real-time Dependable Systems**. In proc. IEEE/ACM COMSWARE'06, New Delhi (India), January 2006.

[6] K. Ravindran, Jiang Wu, M. Rabby, K. A. Kwiat, and A. Sabbir. **Performance Engineering of Replica Voting Protocols for High Assurance Data Collection Systems**. In proc. IEEE/ACM COMSWARE'08, Bangalore (India), January 2008.

[7] C. Koliver, K. Nahrstedt, J. M. Farines, J. D. S. Fraga, and S. A. Sandri. **Specification, Mapping, and Control for QoS Adaptation**. In Journal of *Real-time Systems*, 23, pp.143-174, Kluwer Academic Publishers, 2002.

[8] K. Ravindran and X. Liu. **Service-level Management of Adaptive Distributed Network Applications**. in proc. *Intl. Service Availability Symp.*, ISAS 2004, pp.101-117, Munich (Germany), April 2004.

[9] C. G. Lee, C. S. Shih, and L. Sha. **Online QoS Optimization Using Service Classes in Surveillance Radar Systems**. In *Real-time Systems*, Springer Netherlands, vol.28, no.1, pp.5-37, Oct.2004.

[10] C. D. Gill, J. P. Loyall, R. E. Schantz, M. Atighetchi, J. M. Gossett, D. Corman, D. C. Schmidt. **Integrated Adaptive QoS Management in Middleware: A Case Study**. proc. *IEEE-RTAS*, Toronto (Canada), 2004.

[11] O. Babaoglu. **Non-blocking Commit Protocols**. Chap. 7, *Distributed Systems*, ed. S. Mullender, Addison-Wesley Publ. Co., 1993.

[12] B. Schneier. **Applied Cryptography**. 2nd Ed., John-Wiley Publ., 1996.

[13] M. Castro and B. Liskov. **Practical Byzantine Fault Tolerance.**. In proc. 3rd Symp. on *Operating Systems Design and Implementation*, New Orleans (LA), Febr. 1999.