Impact of WAN Channel Behavior on End-to-end Latency of Replication Protocols*

Roberto Baldoni[1], Carlo Marchetti[1] and Antonino Virgillito[1,2]
[1]Dipartimento di Informatica e Sistemistica "Antonio Ruberti", Università di Roma "La Sapienza"
[2] Istituto Nazionale di Statistica, Roma
email:{baldoni, marchet, virgi}@dis.uniroma1.it

Abstract

Software replication of stateful *services is typically implemented using two-tier architectures, in which clients directly interact with replicas running distributed agreement protocols for ensuring consistency. In general, performance of these protocols is sensitive to network delays, which might consequently reduce service availability. Therefore, in previous works we introduced three-tier software replication, in which agreement protocols run in an apposite tier (detached from clients and replicas) that can be independently deployed in a controlled and stable part of the network.*

In this paper, we analyze the performance of replication protocols implemented using two- and three-tier architectures using a simplified wide-area network model that considers two types of behaviors for channels, i.e., normal *(small and predictable latency variations), and* slow *(high and unpredictable latency variations). This channel model is instantiated using traces of real Internet latencies measured sending HTTP requests to Internet web-sites at varying rates. Then, by exploiting traces, we simulate simplified versions of three replication protocols (i.e., active, passive, and three-tier replication), and we show how the end-to-end latency of each protocol is related to the number of slow channels. Results mainly demonstrate that the availability of a service replicated through a three-tier architecture is less affected from channel slow-downs.*

1 Introduction

Software replication is a well-known class of techniques suitable for increasing the dependability of stateful software services in the presence of failures. During the last twenty years, several software replication algorithms and systems have been proposed in the literature (e.g., [26, 9, 13, 15, 21, 20, 22, 24]) differing, among the others, for the class of tolerated node and network failures (e.g., crashes, omissions, Byzantine, and arbitrary behaviors), for the class of supported replicas (deterministic and nondeterministic), and for the consistency criteria enforced (e.g., lazy replication, linearizability). In general, replication algorithms typically enforce a given consistency criterion by making replicas exchange some synchronization messages among them after service requests. As a consequence, achieving actual high availability and consistency of stateful services replicated using these algorithms requires communication channels to behave in a mostly stable and predictable way, which, along with the absence of multicast primitives, may limit the deployment of replicated stateful services over wide area networks (e.g., the Internet).

Starting from these observations, and in order to reduce the sensitivity to network delays of the performance of replication algorithms, we introduced a novel class of replication algorithms that exploit a three-tier architecture in order to avoid service replicas exchange messages among them [23, 5]. In particular, we have focused on crash-tolerant algorithms enforcing linearizability (or "strong replica consistency"), which necessitate running complex agreement protocols [17]. In three-tier replication, these protocols run in a single network node that is made highly available using software replication techniques *inside of it*, and that controls the state of service replicas using only a reliable unicast communication primitive to communicate with them. This suffices for avoiding replicas directly exchange messages, as in other "classical" crash-tolerant algorithms (e.g. active and passive replication [17]) that we name *two-tier*.

This work complements other studies of three-tier replication, by proposing a performance comparison between two- and three-tier replication algorithms. In particular, in this work we study how the variability of channel delays in wide area networks (e.g., the Internet) influences the end-to-end latency of two algorithms exploiting a two-tier ar-

*This paper has been partially supported by project MAIS, funded by Italian MIUR and by the EU-funded project SemanticGov.

chitecture (namely, active and passive replication), and of a three-tier algorithm.

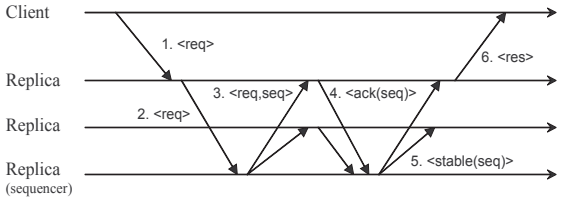
In more detail, our analysis starts from a modeling of algorithms and of the network. Algorithms are modeled by abstracting out from several "local" (e.g., the computations performed within a single network node) and "global" details (e.g., failure detection and group membership protocols). The model then mainly focuses on the patterns of messages exchanged over the wide area network by the algorithms in absence of failures. The network is modeled as a collection of point-to-point channels connecting network nodes characterized by average latencies and their variance. In particular, a *normal* channel presents stable and predictable delays during *any* protocol run, while a *slow* channel may abruptly slow down for an unpredictable amount of time during *some* protocol runs, i.e., it has a heavy-tailed delay distribution [14].[1]. This simplified network model is then instantiated using real Internet latency traces (obtained by experiments run in our labs by getting resources of variable sizes from several HTTP servers deployed over the Internet) that are then classified (by inspection) as those of normal and slow channels (ambiguous traces are discarded). Finally, we run trace-based simulation of the algorithms while varying the number of slow and normal channels. The fairness of the comparison is ensured by the use of the same set of traces for simulating each protocol. This follows from addressing network latency fluctuations within the channel model, which enables reproducing their effects in the same way in all the protocol simulations. The simulation results show an increased resilience of three-tier replication to channel slow-downs.

This paper is organized as follows: Section 2 describes the replication protocols considered throughout this work introducing their models based on simplified message patterns of failure-free runs; Section 3 describes the test-bed used for comparing protocols, consisting in the channel model, and the methods used for instantiating this model over real traces of Internet channels, as well as for performing trace-based simulations of protocols; Section 4 presents the results of our simulations; Section 5 deals with the main existing contributions relating to our work, and Section 6 concludes the paper.

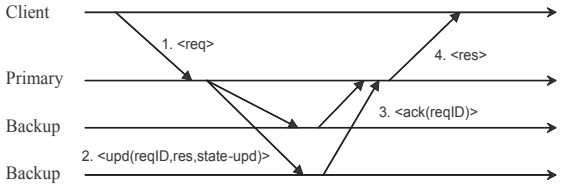
2 Replication Protocols

This section describes the replication protocols considered in this work and introduces their simplified models. All these protocols tolerate *crash failures* (each replica behaves according to its specification until it possibly crashes, i.e., it stops performing any further action), and enforce *linearizability* (or "strong replica consistency", [19]). In the following, we describe two-tier protocols (namely, active and passive replication) first, and then a three-tier protocol supporting deterministic replicas (introduced in [23]), which can be extended to support non-deterministic replicas with no impact on the pattern of messages exchanged with clients and replicas. In all these cases, we assume clients and replicas to communicate through a *reliable unicast* communication primitive.

[1]Note that all real wide area channels are likely to result "slow" if observed for a sufficient amount of time. However, as pointed out in the following sections of the paper, normal channels are a useful abstraction to evaluate sensitivity of distributed algorithms to network delays



(a) Two-tier Active Replication (2TA)



(b) Two-tier Passive Replication (2TP)

Figure 1. Message Patterns of Two-tier Replication Protocols

2.1 Two-tier Replication

In two-tier replication protocols, clients directly interact with replicas that run the same software in distinct network nodes. By cooperating as peer processes, replicas are therefore in charge of enforcing strong consistency before returning replies to clients, which is obtained by running agreement protocols [17]. These protocols commonly use failure detectors [11] and timeouts to detect replica failures and to trigger further protocol phases, which ensure liveness in the presence of failures. The protocol models used for simulations omit all the details related to this complex issue, as we consider only failure-free runs (see Sect. 3.3).

Active Replication. In active replication (or state machine replication [27, 12]) protocols, client requests are executed by replicas in the same order (until a replica possibly crashes). Linearizability of executions is thus guaranteed

as long as replicas are deterministic. In the simplified two-tier active replication protocol model considered throughout this paper (**2TA**) each client issues its requests to a randomly chosen replica, and replicas run a fixed-sequencer uniform total order multicast protocol (implemented over the reliable unicast primitive) to agree on the order of executions.[2].

Figure 1(a) depicts a run of 2TA, where arrows represent unicasts, and numbers identify the sequence of communications steps. A request (req) is issued by a client (Step 1) by contacting a randomly chosen replica that forwards the request to a particular replica, i.e., the so-called *sequencer*, which assigns a sequence number (seq) to req (Step 2). This number is then propagated to other replicas along with the request (Step 3). Then (Step 4) the sequencer waits for acknowledgments (ack messages) from other replicas and notifies them (Step 5) that all acknowledgments have been received (i.e., that they are *stable* messages).The contacting replica, once updated about the stability of the acknowledgments, finally returns the reply back to the client (Step 6).

Passive Replication. In passive replication (or primary-backup [10]) protocols, the order of request executions and the state of all replicas after each of such executions are both determined by a special replica, namely the *primary*, which is the only replica accepting and processing client requests (other replicas are named *backups*). The primary thus performs a serialization of all request executions, as well as of state updates toward backups. Passive replication supports nondeterministic replicas, and linearizability is ensured as long as only one primary replica exists at any point of a logical time (or *view*), which is achieved using a View Synchronous Multicast (VSM) [17].

Figure 1(b) shows a failure-free run of the simplified passive replication protocol model (**2TP**) considered throughout this paper. After having received a client request and computed both the request result and a state update for backups (Step 1), the primary sends an update message to the backups (Step 2) that reply with an acknowledgment (Step 3). Once all the acknowledgments have been received, the primary sends the reply back to the client (Step 4). A new request can be processed only when all updates for the previous request have been acknowledged by the primary to all the replicas. Let us note that, being the order of executions guaranteed by the primary uniqueness (and abstracting out from the protocols following a primary crash) 2TP basically reduces to the primary performing a stability check of the update messages sent to the backups.

[2]Interested readers can refer to [3] for a formal specification of total order multicast semantics and applications to sequencer-based implementations

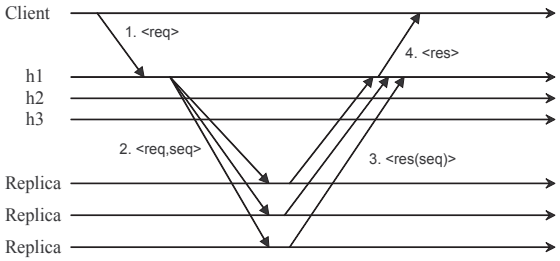


Figure 2. Three-tier Deterministic Replication

2.2 Three-tier Replication

Running distributed agreement protocols among wide area network nodes can be an overkill [2], mainly because of the negative effects on performance of the frequent and unpredictable slow-downs characterizing Internet channels [25, 30]. Indeed, stateful services replicated using two-tier protocols (as well as others with similar communication patterns [14]) can suffer from unavailability periods due to the same protocols sensitivity to channels slow-downs even in absence of crashes [8]. This can be intuitively explained by noting that these protocols require *synchronization phases*, i.e., phases of the protocol code in which at least one replica has to receive a message from all other replicas in order to proceed (Steps 3 and 5 in Figure 1(a), Step 3 in Figure 1(b)). Additionally, WAN channels slow-downs may cause the elapsing of timeouts necessary to detect failures, which might in turn trigger further unnecessary protocol phases and the consequent overhead. Each slow-down of a channel is therefore a potential performance threat of stateful services replicated using two-tier protocols, which has been the main of our motivations for three-tier replication protocols, along with the absence of efficient and diffused multicast primitives in WANs. Another motivation consisted in observing that an efficient fault-tolerant replication service can be easily implemented on a single WAN site hosting several well-connected servers (similar approaches have been used for tackling different problems in [29, 28, 16, 18]).

In the resulting three-tier architecture and protocols, a *middle tier* (i.e., a set of servers connected by channels with low and predictable latencies) is interposed between clients (the *client tier*) and the actual service replicas (the *end tier*). Clients can send their requests only to the middle tier that runs fault-tolerant agreement protocols and forwards client requests to end-tier replicas, along with information sufficient for getting strong replica consistency (i.e., sequence numbers and state updates). End-tier replicas compute results according to the information received, and return replies to middle-tier servers. These servers for-

Channel	1Kb	4Kb
IT1	http://nuxi.iit.unict.it/	http://www.medhoc04.diit.unict.it/home.htm
IT2	http://www.econsoc.unina.it/	http://www.rialc.unina.it/sommario.htm
IT3	http://brett.adm.unipi.it/valutazione/	http://masterelearning.di.unipi.it/
IT4	http://www.etmtraining.polito.it/	http://lambda.di.unito.it/rank2/
USA	http://e-lab.propoint.com/	http://ericae.net/
FR	http://www.education.gouv.fr/syst/orgs6.htm	http://www.csl.sony.fr/
UK	http://www.phy.hw.ac.uk/	http://www.hud.ac.uk/courses/
IL	http://msradio.huji.ac.il/	http://mathphys.haifa.ac.il/

Table 1. URLs of Web Pages used for Channel Measurements

ward back replies to clients just after having received *only the fastest reply from end-tier replicas*. As a result, replicas are not involved in synchronization phases, as well as in failure detection and management protocols, which reduces the sensitivity of end-to-end latency to channel slowdowns (see Sect. 4).

A full three-tier replication protocol supporting deterministic objects appears in [23] that includes formal specifications, the pseudo-code of the algorithm, and its correctness proofs. In this protocol, end-tier replicas process requests in the order determined by unique and consecutive sequence numbers piggybacked onto each client request by middle-tier servers. To achieve this, middle-tier replicas implement an instance of a fault-tolerant *distributed sequencer service* [6, 7]. Replica determinism and ordered executions together ensure the equality of all the responses produced by the end-tier. As a consequence, the middle tier can forward back to clients the first reply received from replicas.

Figure 2 shows the message pattern of the simplified three-tier protocol model (**3T**) considered throughout this paper. As usual, the protocol is initiated by a client issuing a request toward a randomly chosen middle-tier server (Step 1), that invokes the sequencer to obtain a sequence number. We abstract out the details of the protocol run among middle-tier servers as it is assumed to be unaffected by channels slowdowns by construction[3]. After the sequence number has been evaluated, it is sent along with the request to all end-tier replicas (Step 2), which execute the request and store the reply in the order of its sequence number, being thus able to not execute the same request twice and to return results of possible duplicate invocations. The first response returned by replicas to the middle tier (Step 3), is then forwarded back to the client (Step 4).

This protocol can be extended to support nondeterministic objects without modifying the pattern of messages exchanged among clients, middle-tier, and end-tier replicas. It is possible to show that modifications for supporting nondeterministic replicas would impact only on the message contents and on the agreement protocols executed within the middle tier. As a consequence, the simulations results presented in Sect. 4 can be generalized to the protocol version embedding these modifications.

[3]Let us note that [4] presents a performance study of a middle-tier protocol implemented within a local area network.

3 Comparison Test-bed

In this section we detail the general framework we developed for recreating a dynamically changing network environment in which comparing the behavior of the replication protocols we presented above. The characterization of the dynamics of the network is one of the main problems we have to face, since a wide-area network exhibits a wide variety of unpredictable changes, which are difficult to capture into a tractable model which at the same time provides realistic and significant results. The metric on which we focus is the end-to-end latency perceived by clients of the protocols, exploring in particular its fluctuation when network instability is present. We do not consider other characterizing aspects of network channels, such as loss rate and throughput, for reasons explained below. Our comparison framework is composed by three elements, detailed in the remainder of this section:

- A simplified model of Internet channels: it defines a classification of wide-area network channels into two classes (normal and slow), according to their overall behavior.
- Instantiation of the channel model based on actual HTTP traces: a set of real Internet channels is considered, that are classified according to the model, i.e. basing on latency measures.
- Application of the model to the replication protocols: end-to-end latency of the three protocols is computed, simulating the deployment of the protocols over distinct combinations of normal and slow channels. Latency values for each channel are obtained from the HTTP traces.

3.1 Model of Internet Channels

The modeling of Internet channel has been subject of a large amount of studies [25, 30]. Most of these studies recognize the heavy-tailed nature of channel latencies distribution, due to variations of traffic patterns along the day, temporary unavailability of network paths, unpredictable bursts of packet loss and so on. In general, capturing channel behavior within a framework which is at the same time realistic and simple enough to be tractable is a very difficult task.

	Average Latency (ms)					
Res Size (KB)			1			4
Req rate (req/sec)	1	10	100	1	10	100
IT1	30	30	32	48	48	232
IT2	17	17	16	31	31	269
IT3	17	17	19	48	48	640
IT4	23	23	30	40	40	407
USA	776	1104	1497	706	3578	7994
FR	181	126	443	461	659	1300
UK	267	2324	7643	478	294	2921
IL	240	3386	1105	632	2346	2308

	Variance (ms^2)					
Res Size (KB)			1			4
Req rate (req/sec)	1	10	100	1	10	100
IT1	6.6	6.6	92.7	111.8	111.8	305610
IT2	81.8	21.8	24.5	158.3	158.3	11728
IT3	3.6	3.6	10.8	2275	2275	146130
IT4	25.8	25.8	10.2	148	148	79886
USA	$\sim 10^8$	$\sim 10^8$	$\sim 10^8$	154180	$\sim 10^8$	$\sim 10^8$
FR	38748	17967	44388	52751	$\sim 10^8$	$\sim 10^8$
UK	76900	$\sim 10^8$	$\sim 10^8$	$\sim 10^8$	$\sim 10^8$	$\sim 10^8$
IL	16712	$\sim 10^8$	$\sim 10^8$	256690	$\sim 10^8$	$\sim 10^8$

Table 2. Variance of latency distributions of HTTP channels

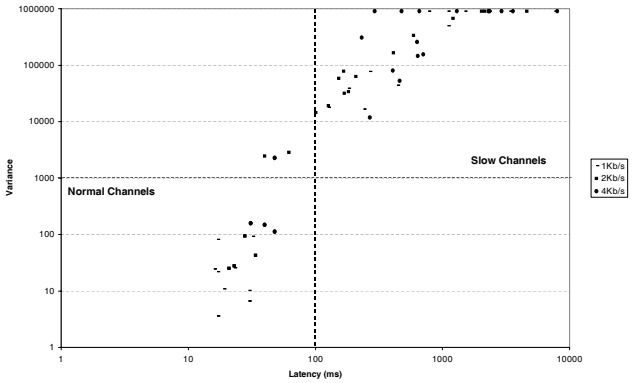


Figure 3. Distributions of Average and Variance of end-to-end Latencies of HTTP Channels

We are specifically interested in determining how latency fluctuations affect the overall latency of replication protocols. In other words, for our purposes, it is not necessary to identify a completely representative model of Internet channel latency but rather one that focuses on latency fluctuations. Then, we introduce a simplified channel model based on a sharp classification into two general classes, according to the behavior of a channel over a given time period T_o (observation period):

- normal channels: average latency during T_o is bounded by $a\bar{v}g$ and the variance is lower than $\bar{v}$. This models channels whose behavior remains stable in all runs of the protocol during the observation time.
- slow channels: average latency is higher than $a\bar{v}g$ and the variance is higher than $\bar{v}$. This models channels that can abruptly slow down in some protocol run or during T_o.

The separation between the two classes (that is, the values of $a\bar{v}g$ and $\bar{v}$) is actually determined by the specific deployment topology and the observation period. Channels that exhibit average latencies lower than $a\bar{v}g$ and variance higher than $\bar{v}$ are excluded from the model. That is, we do not consider the naturally unstable behavior which is proper of Internet channels when observed during a long period. In other words, the fact that a channel is classified as normal or slow means that "most of the time" during the measurement period it behaves as such.

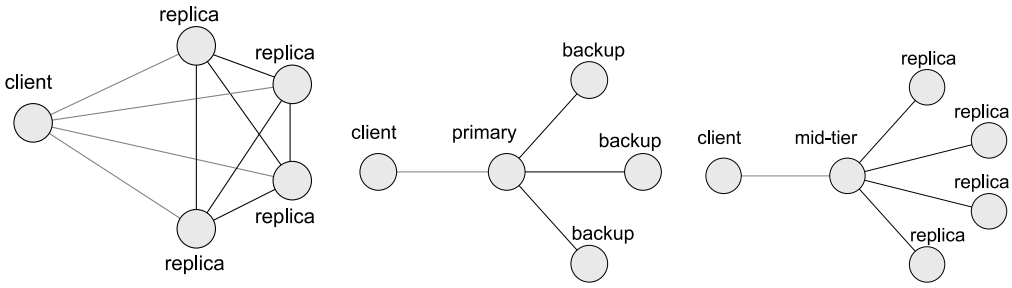
3.2 Instantiating the Model through HTTP Traces

We derived a specific real-world instantiation of the protocol, by considering latency traces obtained by issuing variable-size HTTP requests to servers deployed in different locations over Internet. The deployment of replication protocols over the Internet can be limited by the presence of corporate firewalls filtering out packets referring to custom protocols. Then, in practice, solutions such as HTTP tunneling or standard HTTP-based protocols (such as SOAP) are used for inter-domain communication among applications. Following this observation, in our opinion a HTTP-level measure could better reproduce realistic deployment conditions.

In the following we describe the measurement methodology for obtaining HTTP traces. We developed a custom tool that issues HTTP requests and measures their round-trip time. The tool exploits APIs from the Google search engine, that define methods for locating web pages, given their geographical position and size. This allowed us to vary the size of the HTTP response returned by the queried HTTP servers. Latency is measured at the network interface level, thus excluding effects due to their processing at the operating system and application level.

We considered 8 HTTP servers, placed in different geographical locations (Italy, USA, UK, France, Israel). URLs of the queried web pages are presented in Table 1. For each location, we considered web pages of different size (1, 2 and 4Kb)[4]. Requests start from the measurement tool running in our department in Rome. Measurement period was set to 2hrs. Frequencies of requests was 1, 10 and 100 requests per second, for page sizes of 1, 2, 4Kb. The different measurements are intended to simulate different load conditions on a channel, characterized by the rate and size of messages transiting on the channel at the moment of protocol execution.

[4] All measures and experiments relative to a 2Kb size are not considered in the paper for space motivations



(a) Two-tier Active (2TA) (b) Two-tier Passive (2TP) (c) Three-tier (3T)

Figure 4. Topologies Used in the Experiments

Measurement results are showed in Tables 1 and 2, and plotted in figure 3. Following an observation of the measured values, we decided to set the thresholds for classifying channels into normal and slow as follows: $a\bar{v}g = 100ms$ and $\bar{v} = 10000$ (highlighted in Figure 3). Traces not clearly falling into the two classes (e.g., those corresponding to channels IT3 for 4Kb) are not taken into account in the simulations.

3.3 Protocols Evaluation

Starting from the HTTP traces, we performed an evaluation of the end-to-end latency of the protocols execution. The simulated scenario consists in a set of four replicas invoked by a single client. The connection topologies for the various protocols are depicted in figure 4. In 3T a single node is considered for the middle-tier. In the 2TA protocol the client invokes at each execution a different replica (chosen at random with uniform distribution).

An execution of a protocol is simulated as follows. First, each channel in the protocol (depicted as a thick line in the figure) is associated to the trace of a HTTP channel obtained from the Internet measurements. The parameters of each experiment are the number of slow channels (NS) and the channel load, described by the message rate (MR) and the message size (MS). Four channels are selected from those reported in Table 1 according to the experimental parameters. For example, considering an experiment with $NS = 2$, $MR = 10mess/sec$ and $MS = 4Kb$, the channels IT1, IT2 (normal), FR and UK (slow) might be selected. For the sake of fairness, this set of channels is used for simulating all the three protocols. The end-to-end latency of a single protocol run is computed by considering the message pattern of the protocol and associating a latency value to each communication step. This is done by sequentially picking values from the trace associated to the channel on which the communication step is performed. 1000 runs were executed for each protocol. The model of the protocols abstracts out several details:

- Latency of the communication steps involving the client (depicted as gray lines in the figure) in the end-to-end latency. This is a step common to all protocols.
- Concurrency between clients. Each execution is modeled as isolated from the others and the effect of concurrent protocol invocations is not considered.
- Computation times at servers. Again, this is common to all protocols.
- Latency introduced by the middle-tier computation. As the middle-tier is supposed to be deployed over a LAN, the processing and network latency it introduces is negligible with respect to that of slow wide-area channels. A simulation study, presented in [4], supports our claim showing that the additional latency due to the middle-tier protocols is in the order of 10ms. Hence, in the following the middle-tier is abstracted as a single network node.
- Effect of replica failures.

These simplifications do not affect the generality of our methodology and the significance of the comparison. In two-tier replication the failure of a node involves the expiration of time-outs and the execution of membership protocols, while in three-tier replication failures of replicas are transparent to the middle-tier and do not cause any further latency overhead. Moreover, concurrent executions are delayed in two-tier protocols, that have to complete a synchronization phase before serving a successive request. In 3T, a new request can be issued by the middle-tier as soon as the first replica replies, reducing the delay due to synchronization. Then, by not considering replica failures and client concurrency we are posing ourselves in the best-case scenario mainly for two-tier protocols.

4 Experimental Results

In this section we present the results of our simulations. Experimental results are depicted in Figure 6, showing the average latency and variance of the protocols against the number of slow channels (NS), measured for different values of message rate (MR) and message size (MS). The main focus of our work is the analysis of the effect of the variation of NS over the end-to-end latency. The following observations apply to all cases, regardless of the values of MR and MS. When $NS = 0$, both the average end-to-end latency and its variance are in general lower in 3T wrt 2TA and 2TP. In two-tier protocols the synchronization phase ends when the slowest replica replies, then it introduces a latency equal to the latency of the slowest channel. Differently, the 3T can return a reply right after the fastest replica replies. For this reason, the end-to-end latency is dominated by that of the fastest channel. Results are coherent with this interpretation as the absolute values of latency and variance are conform with those of the channels (i.e., they are always lower than 100ms and 1000 resp.). 2TA results in a higher end-to-end latency than 2TP because it introduces an additional synchronization phase and two communication steps, i.e. Steps 2 and 5 in Figure 1(a)).

When $NS \geq 1$ and $NS \leq 3$ the presence of at least one slow channel, makes the end-to-end latency grow by two orders of magnitude. On the contrary, the end-to-end latency of 3T is in the same order as that with $NS = 0$, regardless of NS (provided one normal channel is present), because the presence of one normal channel ensures a fast response. The same applies to variance, with 3T being almost insensitive to the latency fluctuations of slow channels. When $NS = 4$, the average end-to-end latency of 3T grows by one order of magnitude. This sharp variation is due to the fact that in this situation all replicas reply through a slow channel. However, we recall from Figure 3, that for slow channels the average latency can vary between 100ms and 8000ms. This makes 3T outperform 2T also in this situation. The end-to-end latency of two-tier protocols remains the same when $NS \geq 2$. This happens because end-to-end latency is dominated by the latency of a slow channel which is included in the experiments starting from $NS = 2$.

Finally we comment the effect of MR and MS on end-to-end latency of 3T (Figure 5). MS does not have a significant effect on our measurements, resulting only in a small increment of average latency and variance for 3T. The effect of an increment of MR is more pronounced, causing a general, linear growth of average latencies and variances.

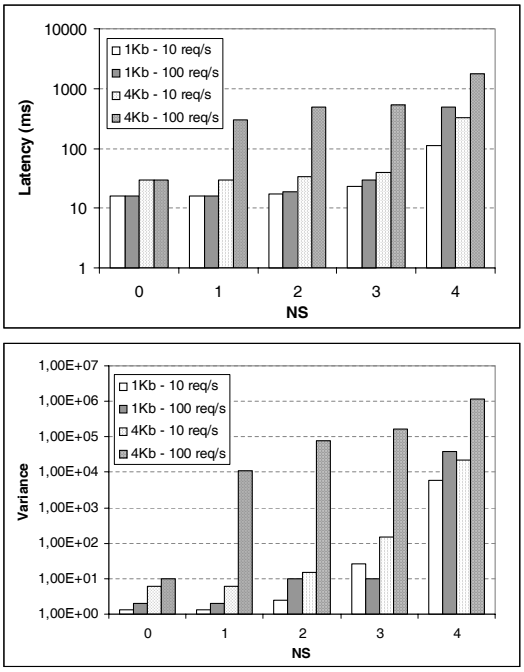


Figure 5. Average Latency and Variance of 3T

5 Related work

Performance of software replication protocols has been analyzed in the literature mainly by presenting or comparing latency and throughput of several group communication protocols (e.g., uniform total order and view synchronous reliable multicast) useful for replicating stateful services while enforcing strong consistency. However, these experiments are commonly performed in local area networks, as only a few group communication systems (e.g., Spread [1]) offer both primitives supporting the implementation of linearizability and the possibility of being deployed over WANs. Further, to the best of our knowledge, no systematic comparison of group communication protocols run over WAN exists, which can be traced back to the sensitivity of these protocols to channel slow-downs ([8]).

Indeed, in [2], Bakr and Keidar first recognize the lack of information about the actual behavior of distributed protocols run over WAN channels, and then they analyze the performance of three simple distributed algorithms (i.e., three simple message patterns) in which a node propagates some information among a set of peers over the Internet. Significantly, this work points out an interesting inversion between experimental results and theoretical metrics (i.e., the number of communication steps). Our work starts from a similar viewpoint, and provides results for the specific software replication protocols analyzed.

Another relevant contribution related to this paper is presented in [14]. In this work, starting from Internet connectivity traces, the authors first derive a model of network unavailability (that can be used to estimate how service avail-

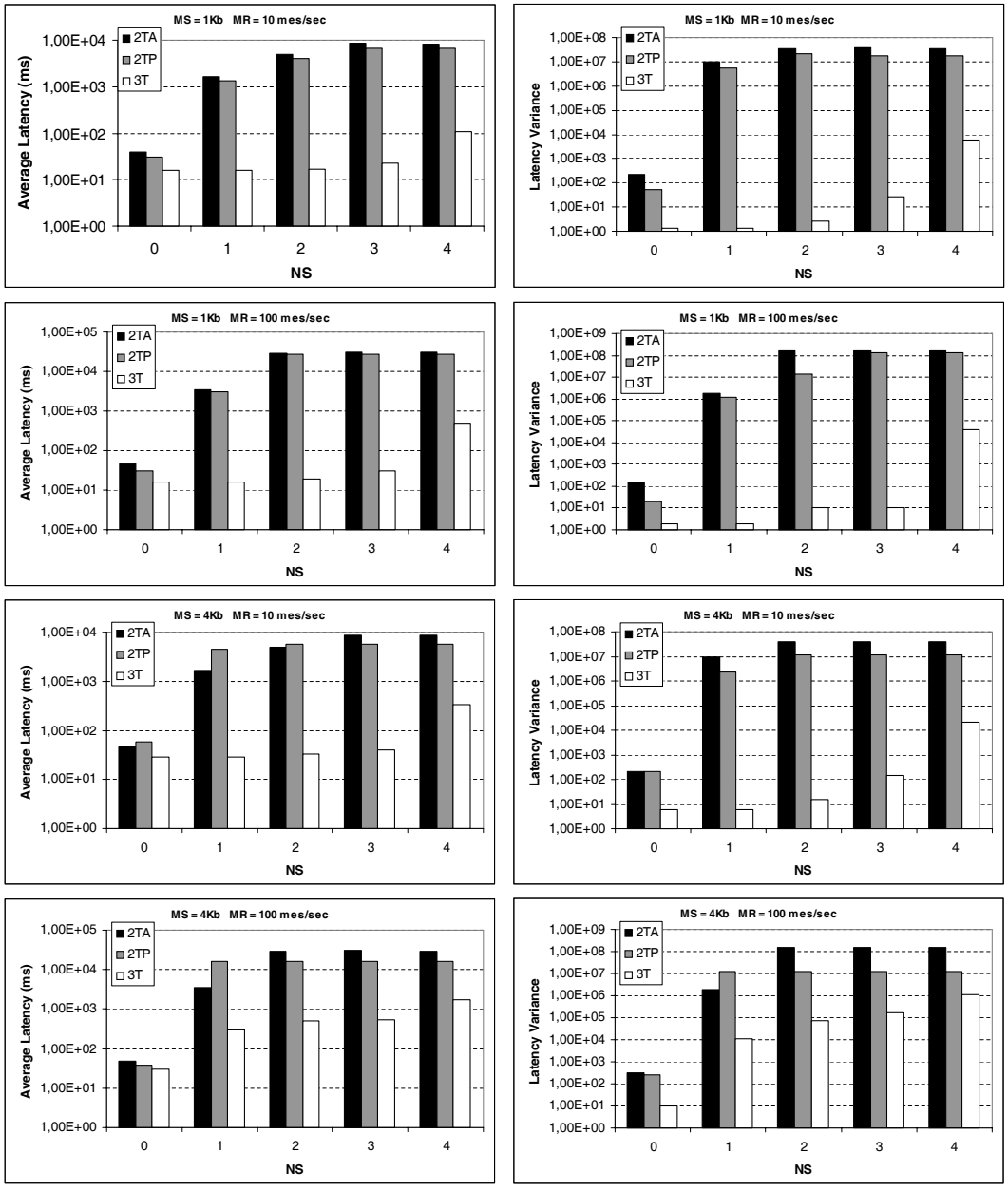


Figure 6. Average (Left) and Variance (right) of Protocols End-to-end Latency

ability might suffer from network failures), and then they analyze the effects on end-to-end service availability of several techniques (caching, prefetching etc.) that can be applied to *stateless* service to reduce the unavailability periods due to network failures. These techniques are not useful for increasing the availability of stateful services if strong consistency is required, which is the topic of this work. However, the network unavailability model can be used to have useful insights about the probability of clients being unable to reach nodes despite service replication.

Finally, due to the simplifications done for obtaining a tractable channel model, this work only marginally relates to the emerging research direction aiming to model the behavior of Internet channels (e.g., [25, 30]), which could be useful to perform a complementary analytic study of the protocols simulated using real Internet traces in this paper.

6 Conclusions and Future Work

In this paper we presented a comparison of end-to-end latency of three protocols for software replications, namely two-tier active and passive replication and three-tier replication. The comparison focuses on the sensitivity of protocols to network channels slow-downs and was carried out through trace-based simulations, using traces obtained from Internet measures. The general results of our experiments are highly favorable for the three-tier protocol that, regardless of the number of slow channels, is able to maintain a stable and low latency.

We are planning to deploy protocols over the PlanetLab infrastructure[5]. However, let us point out that such a study will be complementary to the work presented in this paper. On one hand, PlanetLab will allow to include aspects not considered in our model, such as the effect of concurrent requests and throughput measurements. On the other hand, it is likely that experiments over PlanetLab cannot be repeated exactly under the same conditions for all the protocols. In that sense, our methodology ensures fairness of comparison among the protocols and repeatability of simulations. These two aspects allow us to compare protocols with respect to the very genuine message pattern.

7 Acknowledgments

Authors would like to thank Massimo Cirillo for his help in setting up the simulation environment.

[5] Our department is currently completing the application for joining PlanetLab.

References

[1] Y. Amir and J. Stanton. The Spread Wide Area Group Communication System. Technical Report CNDS-98-4, Center for Networking and Distributed Systems, Computer Science Department, Johns Hopkins University, 3400 N. Charles Street Baltimore, MD 21218-2686, April 1998.

[2] O. Bakr and I. Keidar. Evaluating the running time of a communication round over the internet. In *Proceedings of the twenty-first annual symposium on Principles of distributed computing*, pages 243–252. ACM Press, 2002.

[3] R. Baldoni, S. Cimmino, and C. Marchetti. A Classification of Total Order Specifications and its Application to Fixed Sequencer-based Implementations. *to appear in Journal of Parallel and Distributed Computing*, 6 2006.

[4] R. Baldoni, S. Cimmino, C. Marchetti, and A. Termini. Performance Analysis of Java Group Toolkits: a Case Study. In *Proc. of the International Workshop on scientiFic engIneering of Distributed Java applIcations (FIDJI02)*, pages 81–90, Luxembourg, November 2002.

[5] R. Baldoni, C. Marchetti, and A. Termini. Active Software Replication through a Three-tier Approach. In *Proc. of the 22th IEEE International Symposium on Reliable Distributed Systems (SRDS02)*, Osaka, Japan, October 2002.

[6] R. Baldoni, C. Marchetti, and S. Tucci-Piergiovanni. A fault-tolerant sequencer for timed asynchronous systems. In *Proc. of the 8th Euro-Par Conference*, pages 578–588, Paderborn, Germany, August 2002.

[7] R. Baldoni, C. Marchetti, and S. Tucci-Piergiovanni. Fault-tolerant Sequencer: Specification and an Implementation. In P. Ezhilchelvan and A. Romanovsky, editors, *Concurrency in Dependable Computing*. Kluwer Academic Press, 2002.

[8] K. Birman. A Review of Experiences with Reliable Multicast. *Software – Practice and Experience*, 29(9):741–774, 1999.

[9] K. Birman and T. Joseph. Reliable Communication in the Presence of Failures. *ACM Transactions on Computer Systems*, 5(1):47–76, February 1987.

[10] N. Budhiraja, F. Schneider, S. Toueg, and K. Marzullo. The Primary-Backup Approach. In S. Mullender, editor, *Distributed Systems*, pages 199–216. ACM Press - Addison Wesley, 1993.

[11] T. D. Chandra, V. Hadzilacos, and S. Toueg. The weakest failure detector for solving Consensus. *Journal of ACM*, 43(4):685–722, July 1996.

[12] M. Chérèque, D. Powell, P. Reynier, J.-L. Richier, and J. Voiron. Active replication in delta-4. In *FTCS*, pages 28–37, 1992.

[13] F. Cristian, H. Aghili, R. Strong, and D. Dolev. Atomic Broadcast: From Simple Diffusion to Byzantine Agreement. In *Proc. of the 15th International Conference on Fault-Tolerant Computing*, Austin, Texas, 1985.

[14] M. Dahlin, B. Chandra, L. Gao, and A. Nayate. End-to-end WAN service availability. *IEEE/ACM Trans. Netw.*, 11(2), 2003.

[15] R. Friedman and A. Vaysburd. Fast Replicated State Machines over Partitionable Networks. In *Proc. of the 16th IEEE International Symposium on Reliable Distributued Systems (SRDS)*, October 1997.

[16] R. Guerraoui and S. Frølund. Implementing E-Transactions with Asynchronous Replication. *IEEE Transactions on Parallel and Distributed Systems*, 12(2):133–146, 2001.

[17] R. Guerraoui and A. Schiper. Software-Based Replication for Fault Tolerance. *IEEE Computer - Special Issue on Fault Tolerance*, 30:68–74, April 1997.

[18] R. Guerraoui and A. Schiper. The Generic Consensus Service. *IEEE Transactions on Software Engineering*, 27(1):29–41, January 2001.

[19] M. Herlihy and J. Wing. Linearizability: A Correctness Condition for Concurrent Objects. *ACM Trans. on Programming Languages and Systems*, 12(3):463–492, 1990.

[20] I. Keidar and D. Dolev. Efficient Message Ordering in Dynamic Networks. In *Proc. of the 15th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 68–86, May 1996.

[21] B. Kemme and G. Alonso. A Suite of Database Replication Protocols based on Group Communications. In *Proc. of the 18th International Conference on Distributed Computing Systems (ICDCS)*, May 1998.

[22] S. Landis and S. Maffeis. Building Reliable Distributed Systems with CORBA. *Theory and Practice of Object Systems*, 3(1), 1997.

[23] C. Marchetti, R. Baldoni, S. T. Piergiovanni, and A. Virgillito. Fully Distributed Three-Tier Active Software Replication. *IEEE Transactions on Parallel and Distributed Systems*, 17(7), 2006.

[24] L. Moser, P. Melliar-Smith, and P. Narasimhan. Consistent Object Replication in the Eternal System. *Theory and Practice of Object Systems*, 4(3):81–92, 1998.

[25] V. Paxson. End-to-end internet packet dynamics. *IEEE/ACM Trans. Netw.*, 7(3):277–292, 1999.

[26] D. Powell, G. Bonn, D. Seaton, P. Verissimo, and F. Waeselynck. The delta-4 approach to dependability in open distributed computing systems. In *Proc. of the 18th IEEE Int. Symp. on Fault-Tolerant Computing (FTCS-18)*, New York, USA, June 1988.

[27] F. B. Schneider. Replication Management Using the State Machine Approach. In S. Mullender, editor, *Distributed Systems*. ACM Press - Addison Wesley, 1993.

[28] P. Verissimo and A. Casimiro. The Timely Computing Base model and architecture. *IEEE Transactions on Computers - Special Issue on Asynchronous Real-Time Systems*, 51(8):916–930, August 2002.

[29] J. Yin, J.-P. Martin, A. Venkataramani, L. Alvisi, and M. Dahlin. Separating agreement from execution for byzantine fault tolerant services. In *Proceedings of the nineteenth ACM symposium on Operating systems principles*, pages 253–267. ACM Press, 2003.

[30] Y. Zhang, N. Duffield, V. Paxson, and S. Shenker. On the constancy of internet path properties. In *Proceedings of the First ACM SIGCOMM Workshop on Internet Measurement*, pages 197–211. ACM Press, 2001.