# A Reactive Measurement Framework*

Mark Allman and Vern Paxson

International Computer Science Institute

**Abstract.** Often when assessing complex network behavior a single measurement is not enough to gain a solid understanding of the root causes of the behavior. In this initial paper we argue for thinking about "measurement" as a *process* rather than an *event*. We introduce *reactive measurement* (REM), which is a technique in which one measurement's results are used to automatically decide what (if any) additional measurements are required to further understand some observed phenomenon. While reactive measurement has been used on occasion in measurement studies, what has been lacking is *(i)* an examination of its general power, and *(ii)* a generic framework for facilitating fluid use of this approach. We discuss REM's power and sketch an architecture for a system that provides general REM functionality to network researchers. We argue that by enabling the coupling of disparate measurement tools, REM holds great promise for assisting researchers and operators in determining the root causes of network problems and enabling measurement targeted for specific conditions.

## 1 Introduction

Because networks are vast collections of integrated components, it can often be the case that analyzing some network behavior in depth (for characterization, tuning, or troubleshooting) requires adapting on-the-fly what sort of measurements we conduct in consideration of the conditions manifested by the network. While the technique of adapting measurements dynamically has been recognized by practitioners in a number of contexts, a key missing element has been the ability to tie together disparate forms of measurement into a cohesive system that can automatically orchestrate the use of different techniques and tools.

To this end, we outline a new measurement paradigm: *reactive measurement* (REM). The vision of REM is to provide a platform that can *couple* measurements—both active and passive—together in a way that brings more information to bear on the task of determining the root cause of some observed behavior. For instance, consider the problem of analyzing the failure of a web page to load. When a REM system observes unsuccessful web page requests, it can automatically execute a set of diagnostic measurements designed to winnow the set of possible reasons for the failure down to the root cause(s) (e.g., a subsequent *traceroute* may highlight a disconnect or loop in the path). While any particular reactive measurement task can be manually pieced together with straightforward scripting, many of the tasks (collecting events, expressing dependencies, managing timers, archiving results to varying degrees) benefit a great

* Passive and Active Measurement Conference, April 2008.

deal from a "toolbox" approach. Essentially, it is the absence of such a toolbox that, we believe, has led to a failure to exploit reactive measurement to date.

The basic notion behind the reactive measurement paradigm is that automatically coupling disparate measurement techniques can bring more information to bear on the task of gaining insight into particular network behavior. The fundamental REM building block is having one measurement's result trigger additional *reactive measurements*. Thus, when a particular behavior is observed, we can automatically trigger additional measurements to work towards determining the root cause(s) of the behavior. Furthermore, as those tools hone in on the underlying reasons—or determine that a given hypothesis is incorrect—their output can again trigger the additional measurements needed to drive progress forward. The paradigm of reactive measurement is to think of "measurement" as a process rather than a simple activity. The goal of the process is to gain insight, and in a system as complex as today's networks such a task will likely involve more than one assessment technique.

This quite simple idea holds promise both for providing a foundation for significant advances in network troubleshooting, and for fostering new types of Internet measurement studies. Regarding this latter, the literature is filled with Internet measurement studies that evaluate the behavior of networks that are working as expected. These studies sometimes offer glimpses of the failure modes present in the current network, when such glitches are observed in the course of taking measurements (e.g., [12] identifies routing "pathologies", which are then removed from subsequent analysis). REM, however, enables the opposite approach. Because REM can key on *anomalies* in the network, REM can be used to trigger measurement infrastructure precisely when unexpected events occur, enabling us to learn a wealth of information about the causes of the problems and their immediate effects. We can further ultimately envision REM as the basis for networks that can automatically diagnose problems and take steps to work around detected failures.

REM enables fundamentally new ways to measure network behavior that cannot be accomplished with stand-alone active or passive techniques. Consider the case of measuring failures in the Domain Name System (DNS). While a number of studies on the operation of the DNS have been conducted (e.g., [6]), the fundamental question "how long does a particular DNS failure persist?" remains largely unanswered. This question cannot be answered by simply monitoring the network, because the experiment is then beholden to users who may or may not trigger additional DNS requests after a failure (particularly if they've been trained by the failure patterns they've experienced in the past). Alternatively, researchers could actively query the DNS for a set of hostnames independent of the requests invoked by actual users. In this case, following up on failed requests is straightforward. However, while this approach can shed light on the original question, the workload imposed on the DNS and network is synthetic and likely unrealistic. Using reactive measurement allows for bringing both active and passive measurements to bear to answer the basic question: a monitor can observe naturally occurring DNS requests in the network, and, upon noticing a failed DNS request, the REM system triggers an active measurement tool to periodically query the DNS to determine how long the failure persists, whether the failure is intermittent, etc. We can also invoke additional tools to determine *why* the DNS requests are not completing.

A second use of REM is for *targeting* measurements. Consider a packet-trace study investigating the behavior of networks and protocols under "very congested" conditions to gain insight into how to evolve protocols and algorithms to work better in such situations. The way this is often done today is to trace the network for a lengthy block of time and then post-process the resulting traces for periods when the network is "very congested," discarding the remainder of the trace. This methodology is scientifically sound, but logistically cumbersome due to the volume of traces that must be initially collected. Using a REM system, however, the researcher could first passively assess the state of the network, and then trigger detailed packet capture only when the network is in the desired state. In this way, not only does the researcher not have to capture and store traffic that will ultimately not be used, but the traffic that is captured is immediately available for analysis without pre-processing. In this case, REM does not provide a methodology for conducting a fundamentally different experiment than could otherwise be undertaken (as is the case for the DNS investigation described above), but it eases some key logistical challenges by providing targeted measurements. *This is not a minor benefit, as the logistical burdens can easily be such that they, in fact, provide the ultimate limit on how much useful data is gathered.*

Finally, we note that while we have framed the REM system in terms of reactive *measurements*, the system is general enough to support a much broader notion of a *reaction*—such as something that is executed, but is not a measurement. For example, a generic reaction could page a network operator when the system has determined that a router has crashed. Ultimately, the REM system could be used as a platform to automatically *mitigate* or *correct* observed problems. For instance, if the REM system determines that a local DNS server has crashed, it could trigger a backup server to take over (as well as notifying operators of the change). Using the REM framework in this way offers great potential for providing a powerful method to add robustness to networks.

The remainder of this paper is structured as follows. We sketch related work in § 2. In § 3 we present the architecture of a prototype REM system that we have developed to support diverse measurement needs by providing the "glue" with which to tie together arbitrary active and passive network measurement tools. We briefly summarize in § 4.

## 2 Related Work

First, we note that the wealth of work the community has put into developing active and passive measurement tools forms a *necessary* component of the REM framework. As outlined in this paper, the reactive measurement system conducts no measurements itself. Rather, it leverages the results from independent active measurement tools and passive traffic monitors as input into a decision process as to what subsequent measurements are required to uncover the cause(s) of a given network phenomenon.

Many past studies have employed multiple measurement techniques in an attempt to gain broader insight on a particular problem than can be obtained when using a single measurement method. For instance, [9] uses both *traceroute* and BGP routing table analysis to determine the AS path between two given hosts. The key difference between these kinds of studies and the REM framework outlined in this paper is in

REM's *automated coupling* of measurements. REM specifically defines dependencies between the output of a measurement tool and what (if any) additional measurements are required. We note that REM is orthogonal to and does not obviate the usefulness of studies like [9] that leverage information from multiple independent measurements.

The literature also has examples of researchers utilizing the reactive measurement notion. For instance, [3] uses *traceroute* measurements to followup on the detection of possible "missing routes" found by analyzing BGP routing tables. Another example is discussed in [2], whereby incoming email is first classified as spam or ham and then the URLs within the spam are followed in an effort to characterize various scams. While researchers have used REM techniques in the past for specific purposes, what has been missing is to systematize these mechanisms in order to make REM broadly available to the research community as a general approach.

In addition, we note that our framing of measurement as a process rather than an event shares some properties with PDA [5] (which is mainly focused on host problems, but does touch on connectivity issues as well), ATMEN [8] (which is largely concerned with coordinating distributed triggered measurements across organizations), and the general idea of "trap directed polling" via SNMP information. All of these systems in some fashion make use of one measurement to drive another measurement (and/or ultimately make a conclusion), but all focus on different aspects of the problem.

Reactive measurement shares some of the goals of the "knowledge plane" (KP) proposed in [4]. The KP envisions continuously gathering information about the network. When particular behaviors need further investigation the KP can be queried to gain a breadth of relevant information. One immediate and practical problem with the KP approach is the immense task in gathering and sifting through information about the entire network. REM proposes essentially the opposite approach: rather than synthesizing from already-gathered information, REM aims to *adaptively* gain insight into particular observed behaviors by running a series of measurements in response to a given phenomena. REM thus has the advantage that it can be conducted *locally*. No distributed data substrate—with the attendant difficulties of scaling, privacy, security, trustworthiness—needs to be constructed. That said, we note that REM in some sense is also orthogonal to ambitious approaches such as KP. The two could be coupled, such that facts learned by REM activity are fed into the KP data substrate, and REM itself could incorporate facts extracted from the substrate to drive its local decision process (as discussed in more detail in § 3.4).

Finally, we note that intrusion detection systems (IDS) share some high-level notions with REM [18, 13]. IDS systems passively observe traffic to draw observations regarding network activity. These observations can be hooked to a "reaction", ranging from logging an event to resetting a TCP connection to adding a firewall rule to block traffic from a host that is port scanning the network. The REM concept of a reaction is much broader than the security-related reactions that popular IDS systems incorporate. In addition, IDS systems offer a passive view of the network, while reactive measurement allows for active probing to determine the state of the network. However, the ability of some IDS's to sift through large traffic streams to find specific types of high-level activity offers great promise of leverage within the REM framework (see § 3).

## 3 REM Architecture

This section presents an architecture for a generic, reusable reactive measurement system suitable for a broad array of measurement efforts. Our aim is to both explicate the approach and solicit input from the community while the effort is in its formative stages. We begin with a discussion of incorporating external measurement tools into the system. We then present the internal machinery that drives the measurement procedures, briefly delving into some of the details. Finally, we discuss possibilities for integrating the REM system with other external resources.

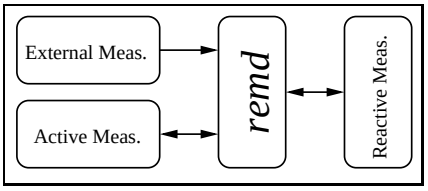


**Fig. 1.** Conceptual layout of the reactive measurement system.

### 3.1 External Interactions

Fundamentally, the REM system couples measurements with reactions. Figure 1 illustrates the system's basic structure: arbitrary measurement tools glued together using a daemon, *remd*, that can be run on any general purpose computer connected to the network to be measured. *remd* provides an interface to and from traditional measurement tools, as well as a method for specifying the relationships between the measurements (outlined in the next subsection). First, we outline the various measurements shown in Figure 1 with which *remd* interacts:

- **Active Measurements**. *remd* can initiate independent active measurements based on a run-time-configured schedule and incorporate their results as input into whether or not to follow up with a reactive measurement, and in what form. For instance, a simple *ping* measurement may be executed every $N$ seconds, with various pieces of information returned (e.g., success/failure, loss rate, presence of reordering, etc.) to *remd*. These results could then trigger additional measurements in an attempt to determine the reason behind the initial observations.
- **Reactive Measurements**. These are measurements that *remd* executes in response to previously-measured network phenomenon. For instance, if a tool reports to *remd* that the loss rate between the local host and a given remote host exceeds a threshold, then a reactive measurement can be triggered to attempt to determine the cause of the increased loss rate or how long it persists. The results of reactive measurements are fed back to *remd* and are then used to determine whether further reactive measurements are needed. Reactive measurements can be active or passive measurements.

– **External Measurements**. These are measurement results delivered to *remd* without *remd* initiating them itself. These measurements could come from SNMP monitoring systems, routers, intrusion detection systems (IDS), system log analyzers or custom built monitors. Each of these entities potentially has a unique and useful vantage point from which to assess certain network conditions and attributes.

The various components of the system interact by passing structured messages between the *remd* and the measurement tools. We can incorporate arbitrary tools into the system by writing simple wrapper scripts[1] that ($i$) understand and process requests formed by the *remd*, ($ii$) evaluate the output of the given tool(s) (return codes, output files, standard output), and ($iii$) form responses in the format *remd* requires. We use XML for requests and responses to ensure an extensible message structure that can accommodate communication with arbitrary measurement tools (their diverse set of arguments and result types). In addition, XML parsers are widely available allowing users to construct wrapper scripts without building complicated parsers and in a wide variety of languages. Finally, we note that while the contents of the messages passed between the *remd* and the various measurement tools must be well-formed, the meaning of the information and its relationship within the overall experiment is defined at run-time by the *remd* configuration, allowing a great degree of flexibility and leaving *remd* as neutral glue.

### 3.2 Internal Architecture

Internally, the REM system has three basic components: a measurement scheduler, an event receptor, and a state machine to capture the linkages between measurements. The measurement scheduler runs measurement tools at prescribed times. For instance, the user may want to run a simple measurement to assess a path periodically, along with successive reactive measurements as dictated by the results of the first measurement. Or, upon detecting a failure the user may wish to run the reactive measurements after a given amount of time, rather than immediately (e.g., to test DNS resolution $N$ seconds after observing a failed lookup). The event receptor receives notifications from external monitors (e.g., an SNMP monitor) that then may initiate a chain of reactive measurements, and from the activity of the reactive measurements themselves. Finally, the state machine manages the transitions between various measurements.

Figure 2 gives an example of an REM state machine. It codifies that REM should start a *ping* measurement based on an internal timer. Based on the results of the *ping* measurement, REM will execute zero, one, or two reactive measurements. If the loss rate measured by *ping* exceeds a threshold $T$, REM executes *treno* [10] in an attempt to determine where in the path the congestion occurs. If the *ping* measurement observes packet reordering on the path, REM uses *cap* [1] to assess the impact of reordering on TCP's congestion control algorithms. Note: if the *ping* indicates a loss rate that exceeds $T$ and packet reordering is present both *treno* and *cap* will be executed (bringing up a number of coordination issues that we discuss in more detail below). In the case where

[1] The NIMI measurement infrastructure [15, 14] has successfully used a similar wrapper script technique to incorporate arbitrary measurement tools.

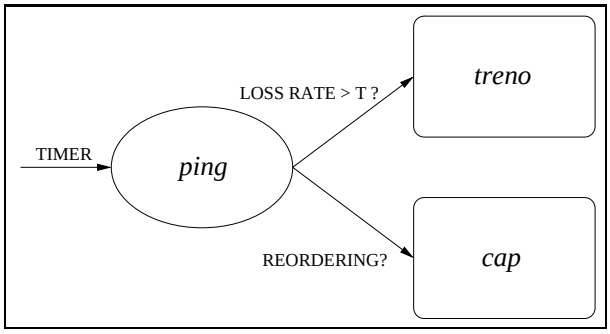


**Fig. 2.** Simple state machine whereby all measurements are invoked by *remd*.

the *ping* measurement indicates both a loss rate below $T$ and no reordering, then no further measurements are executed. In other words, an implicit terminal state follows each state in the machine. If, after executing a measurement, none of the transitions are valid, then the current measurement chain ends. Finally, the *treno* and *cap* states could, of course, also have transitions to additional measurements.

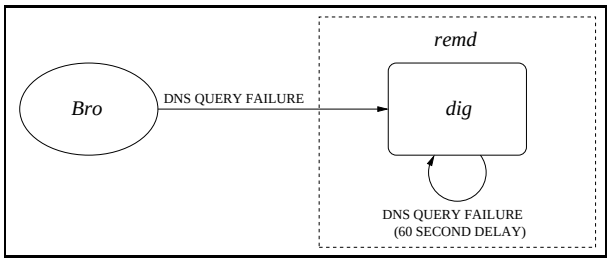


**Fig. 3.** Simple state machine of a reactive measurement triggered by an external monitor.

Figure 3 gives a second example of an REM state machine. Here, *remd* (everything within the dotted line) receives a DNS failure notification from an external source, namely an instance of the *Bro* IDS (which can perform extensive, application-layer analysis of traffic). Upon receiving the message indicating a DNS failure occurred, *remd* executes a *dig* measurement in an attempt to resolve the given hostname. Each time the given hostname cannot be resolved, REM schedules another *dig* measurement for 60 seconds into the future. In addition to setting a time between measurements, a maximum number of attempts can be configured. For instance, inserting 60 seconds between DNS queries and running a maximum of 10 queries may suffice for a given experiment. Of course, a simple periodic timer will not suffice for all situations; our prototype REM system also provides Poisson-based intervals and exponential backoff.

The above examples are clearly simplistic, and the thorny problem of measurement scheduling and collision remains. A user may wish to have two reactive measurements run in parallel in one instance, and serially in another. In addition, a user may wish to base a reaction on the output of multiple measurements. These situations greatly complicate state machine construction. While this complexity can be hidden from the user by providing a high-level interface from which the system then creates the actual state machine, we may need a more powerful abstraction to cover all possible cases in the future. For example, we could use the quite general framework of Petri Nets [16] to codify the reaction path. Alternatively, we could directly employ Bro's events and timers. Our current prototype is based on a simple state machine. As we explore the sorts of reactive measurements we find we want to express in practice we will look to enhancing the system's abstract model to support these sorts of richer couplings.

As indicated above, external notifications to the REM system can come from any network monitoring system (IDS, SNMP, custom developed, etc.). Attempting to interface *remd* with legacy systems may require a lightweight shim to provide the necessary "plumbing". For example, consider integrating the *Bro* IDS into the REM framework. Since the *Bro* system includes a client library for transmitting *Bro* events and typed values, to integrate it with REM we can devise a simple event receiver that understands *Bro* events and translates them into *remd* notifications.

Note, as discussed thus far, the REM system has no particular provisions for security mechanisms over and above those placed on the user-level tools by the underlying operating system. We believe this is the generally correct model. However, we clearly must require access control for external notifications. A natural approach for doing so would be to layer such notifications on top of SSL connections in order to leverage SSL's authentication capabilities. We could potentially augment this with an authorization capability allowing a researcher to define which external monitors can communicate with *remd* and what sort of messages they can send.

### 3.3 Details

The high-level architecture sketched above is realized through a system whereby each experiment keeps a variable list that can be arbitrarily populated with state information by the experiment configuration and the measurement tools as they are executed. For instance, a measurement tool's argument list can be populated by the configuration setting variables for the tool. Wrapper scripts consult the variable list and add to it information about the outcome of a particular measurement. Once a measurement is finished and the updated list returned, *remd* executes the transitions, which are specified using arbitrary Python code that runs in the context of a given variable list. Using this scheme the *remd* is only required to manage the overall measurement *process* and not have any understanding of the measurements themselves. Thus, *remd* is charged with tasks such as moving variable lists around, executing transition code from the configuration, managing processes, and stopping processes that take too long.

### 3.4 Interfacing to External Resources

A REM system such as described above would provide a solid foundation for conducting fundamentally new and different measurement studies. However, the system can be more useful still if it were to contain the ability to interact with different types of external resources. Below we sketch two possibilities.

*Measurement Infrastructures.* Often we can derive more information about a network anomaly by probing the network from multiple vantage points. For instance, a DNS failure may be a local problem to a given network or a more general global problem with one of the root DNS servers. If we perform DNS lookups at only one point in the network (i.e., where *remd* is running), we can fail to observe the full scope of the problem. However, by running the same DNS query from a number of distributed points in the Internet, a more complete story about the failure might emerge. Thus, we should aim to interface the REM system with distributed measurement systems such as *scriptroute* [19] or DipZoom [17]. Such interfaces provide the ability to run reactive measurements at many points in the network simultaneously to gather as much information as possible about network anomalies. Note that by using wrapper scripts, the REM system can accommodate such interfaces without any particular extensions to the general framework: we simply write wrapper scripts invoked by *remd* that, for example, execute *scriptroute* tools to run measurements on alternate hosts and gather the results.

*Measurement Repositories.* While reactive measurement offers a great deal of power, one deficiency is that sometimes the overt trigger for a failure or anomaly comes *late*: that is, by the time we observe the problem, we may have missed valuable precursors that shed light on the problem's onset. We envision a partial counter to this problem in the form of interfacing to measurement repositories. For instance, wrapper scripts could interface with the bulk packet recorder outlined in [7] in an attempt to try to build understanding about the precursors to some observed phenomena. Another obvious source of information could be the RouteViews repository [11] of advertised routing tables.

## 4 Summary

Our two major—if preliminary—contributions are ($i$) developing the general notion of *reactive measurement* as a paradigm that focuses on a measurement *process* as the key to better understanding observed behaviors, and ($ii$) the design and prototyping of a reactive measurement system to aid researchers in using the technique in their own work. We believe that if the community absorbs and leverages this concept in their experimental designs, it can lead to significant advances in better understanding network behavior. We hope by exposing our initial design to the community we will get feedback on important aspects to include in future versions of our framework.

## Acknowledgments

The ideas in this paper have benefited from discussions with a number of people including Fred Baker, Ben Chodroff, Scott Shenker and Randall Stewart. This work was

funded in part by Cisco Systems and NSF grants ITR/ANI-0205519 and NSF-0722035. Our thanks to all.

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors or originators and do not necessarily reflect the views of the National Science Foundation.

## References

1. Mark Allman. Measuring End-to-End Bulk Transfer Capacity. In *ACM SIGCOMM Internet Measurement Workshop*, November 2001.
2. David S. Anderson, Chris Fleizach, Stefan Savage, and Geoffrey M. Voelker. Spamscatter: Characterizing Internet Scam Hosting Infrastructure. In *Proceedings of the USENIX Security Symposium*, August 2007.
3. Di-Fa Chang, Ramesh Govindan, and John Heidemann. Exploring The Ability of Locating BGP Missing Routes From Multiple Looking Glasses. In *ACM SIGCOMM Network Troubleshooting Workshop*, September 2004.
4. David Clark, Craig Partridge, J. Christopher Ramming, and John Wroclawksi. A Knowledge Plane for the Internet. In *ACM SIGCOMM Workshop on Future Directions in Network Architecture*. August 2003.
5. Hai Huang, Raymond Jennings III, Yaoping Ruan, Ramendra Sahoo, Sambit Sahu, and Anees Shaikh. PDA: A Tool for Automated Problem Determination. In *Proceedings of USENIX Large Installation System Administration Conference (LISA)*, November 2007.
6. Jaeyeon Jung, Emil Sit, Hari Balakrishnan, and Robert Morris. DNS Performance and the Effectiveness of Caching. In *ACM SIGCOMM Internet Measurement Workshop*, November 2001.
7. Stefan Kornexl, Vern Paxson, Holger Dreger, Anja Feldmann, and Robin Sommer. Building a Time Machine for Efficient Recording and Retrieval of High-Volume Network Traffic. In *ACM Internet Measurement Conference*, 2005.
8. Balachander Krishnamurthy, Harsha V. Madhyastha, and Oliver Spatscheck. ATMEN: A Triggered Network Measurement Infrastructure. In *Proceedings of WWW*, May 2005.
9. Zhuoqing Morley Mao, Jennifer Rexford, Jia Wang, and Randy Katz. Towards an Accurate AS-Level Traceroute Tool. In *ACM SIGCOMM*, August 2003.
10. Matt Mathis. Diagnosing Internet Congestion with a Transport Layer Performance Tool. In *Proceedings of INET '96*, June 1996.
11. University of Oregon RouteViews Project. http://www.routeviews.org.
12. Vern Paxson. End-to-End Routing Behavior in the Internet. In *ACM SIGCOMM*, August 1996.
13. Vern Paxson. Bro: A System for Detecting Network Intruders in Real-Time. In *Proceedings of the 7th USENIX Security Symposium*, January 1998.
14. Vern Paxson, Andrew Adams, and Matt Mathis. Experiences with NIMI. In *Proceedings of Passive and Active Measurement*, 2000.
15. Vern Paxson, Jamshid Mahdavi, Andrew Adams, and Matt Mathis. An Architecture for Large-Scale Internet Measurement. *IEEE Communications*, 1998.
16. J. L. Peterson. *Petri Net Theory and the Modeling of Systems*. Prentice Hall, 1981.
17. M. Rabinovich, S. Triukose, Z. Wen, and L. Wang. Dipzoom: the Internet measurements marketplace. In *9th IEEE Global Internet Symp.*, 2006.
18. M. Roesch. Snort: Lightweight intrusion detection for networks. In *Proceedings of USENIX LISA*, 1999.
19. Neil Spring, David Wetherall, and Tom Anderson. Scriptroute: A Public Internet Measurement Facility. In *USENIX Symposium on Internet Technologies and Systems (USITS)*, 2003.