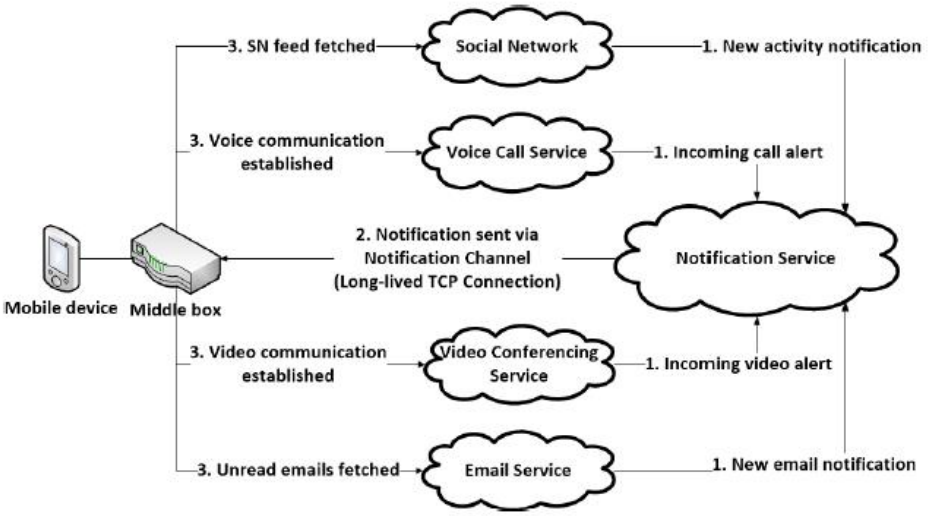
使用自适应的心跳频率来保持 TCP 长连接

概要——在这篇论文中，我们对动态的调整 TCP 长连接中的心跳或者是发送保活数据包的间隔大小给出了一些建议，特别针对的是在移动平台上的推送通知的服务。当一个连接在 NAT（网络地址转换器）（或其他的中间件）上的客户端和一个服务之间的 TCP 连接长时间空闲时，这种空闲有可能是因为 TCP 连接超时而断开连接，为了保持连接不会断掉，在空闲时，客户端设备要通过这个连接发送一些用于保持连接的数据包。为了减少资源占用，这个用来保持长连接的的数据包发送的时间最好在 NAT 连接超时时间内尽可能的长。我们把这样的间隔称为最佳间隔（Optimal Keep-alive Interval）。由于不同网络设备的不同网络设置，在不同网络中，最佳间隔不是一致的。因此，在不同网络中，心跳率将是动态变化的。我们建立一套重复探测技术，这里将他们命名为折半查找（binary）、指数查找（exponential）和复合查找（composite），用来动态的找出最佳间隔；在这个过程中，是由客户端设备来持续改进这个间隔的。我们也对性能极限情况进行了分析。据我们所知，我们是第一个系统的研究这几项技术来动态改进连接间隔的。最后，我们在仿真器上进行了实验，同时也在真实的 Android 设备上实现了这个技术，演示并证明这个方案。

第一章 介绍

智能手机，平板电脑和其他的掌上电脑都尽可能提供给用户最新的数据。包括用户的邮件，新闻广播等。像语音和视频通话一样的实时通信，也使用这些设备。然而由于提到的这些设备电池电量有限，它们不能频繁的轮询来更新数据及进行来电通知等。它们又依赖于被推送服务器推送过来的更新通知。包括 iPhone，Android，Windows Phone，Black Berry(黑莓)等，各移动平台提供了一个通知服务，可以抽象建模如图一所示。



图一 在不同移动平台通知服务模型

这并不是轮询不同的服务去检查是否需要下载数据，而是用户的设备仅仅与通知服务维持一个 TCP 连接。这个连接被叫做通知信道(Notification Channel)。当用户的其他网络服务需要发送最近的活动信息，它便发送一个新的活动通知给通知信道。根据这个通知，设备连接到其他网络，下载新的信息，并关闭连接。对于下载未读的邮件、接收语音和视频的实时通知消息等，也是类似的流程。因此，为了使得看起来似乎与其他服务一直连接的场景，设备必须一直保持通知信道是活跃可用的，甚至在省电模式下也要活跃可用。大多数情况下，这个服务是在后台静默的运行，只有在服务有更改要通知给用户时，才会激发网络访问。当一个设备连接到 NAT、防火墙或者是其他任何中间件(middle-box)时，都可能会造成连接超时。这意味着在可能在一个指定的时间中间这个连接都没有数据传输时，这个连接可能被中间件切断。

研究表明，在商用的家庭网络状况下，NAT 连接超时时间变化巨大[1]。为了保证不会连接超时，空闲时设备需要周期性的与另一端进行通信。这被叫做保活[2](keep-alieve)或者是心跳(heartbeat)，这个间隔被叫做保活或者简称为 KA，为了平衡电池的使用情况和保持这个连接所必须的开销，在连接超时时间内，发送保持连接的保活数据包间隔时间要尽可能的长。这个间隔被称为保持连接的最佳间隔。这个间隔需要根据设备所在的网络环境进行测试后进

行估算。

在这篇论文中，我们提出了几种间隔探测方法来动态改进 TCP 长连接中的 KA 间隔。这些方法包括折半查找，指数查找和复合查找。复合查找综合了折半查找和指数查找的技术不同点。我们在理论方面进行验证并提出建议，通过实际来验证他们。我们给出了若干在不同间隔的情况下，KA 消息最少发送数目的结果。

这篇论文其他部分内容如下，在第二部分回顾了我们早期曾做过的一些相关工作；第三部分描述了几个动态改变 TCP 连接 KA 间隔的技术；第四部分分析了这些技术的几个属性的变化范围；第五部分描述了这些仿真实验及实验结果；第六部分总结这篇论文。

第二章 相关工作

我们分为四个部分组织并进行了文献查询。首先，我们查看了一个研究，这个研究表明，在不同网络环境中，TCP 连接超时范围变化很大，这表明了定时发送 KA 包来维持连接的重要性。然后我们查看了在已经存在的系统中 KA 包发送的情况。我们也查询了一些关于 KA 间隔对于电量消耗的影响的文献。这表明，在尽可能情况下检测和使用更长的 KA 间隔的必要性。最后我们回顾了我们关于使用交互式的方法来测量不同网络参数的所做的一些更早的工作。

A.在 NAT 连接超时的研究

Hatonen et al 在[1]实验中分析了不同家庭网络环境下 TCP 连接超时的情况。被观测到的最短的超时时间间隔是 4 分钟，中值为 1 小时。尽管互联网工程任务组(IETF)建议时间为 124 分钟[3]，超过 50%的设备不遵从这个标准。

另一方面，有一些设备会保持这个 TCP 连接相当长的时间，有时在无通信时超过 24 小时仍未超时。由于连接超时变化范围太广了，一个手机设备在真实网络环境中可能也需要进行测试来找出更长的 KA 间隔并且使用这个间隔来发送 KA 包。尽管想 NSIS[4]或者是 SIMCO[5]这样的协议中确实包含中间件超时的时间说明，然而现在手机经销商通常并不支持这些协议。

B.已经存在的系统上的 KA

Exchange ActiveSync(EAS)协议的的直推功能使用 HTTPS 请求来与服务维持通信[6]。假如服务器上没有数据更新，每个请求都会在服务器上停留一段时间，这个时间由请求者确定，假如时间到了，服务器返回“200 OK”。然而如果底层网络连接超时了，那么就没有返回。客户端过一段时间就会发送另外的 HTTPS 请求。可以在[7]中找到关于 KA 间隔大小的详细讨论。

Android，iPhone，Windows Phone 等平台在客户端和服务器之间会维持一个长连接[8],[9],[10],[11]。这几个系统采用定期与服务器交换信息并且有机制来动态调节 KA 间隔以便获取更好的电池使用表现。在微软公司的专利中，关于客服端和服务器经由的中间件[12],[13]，可以看出在试着使用动态调整 KA 间隔的相关专利内容。然而，这个关于如何选择间隔的精确的策略在专利中没有提及。

在安全传输层协议[14]中的心跳[15]也被用来保证对方是活跃可用的。

C.KA 间隔对于耗电量的影响

Haverient et al 在[16]中说明了，需要特别关注发送 KA 消息的频率对于电池使用寿命的影响。在不同的 3G 和 2G 网络中，他们进行了真实的电量使用情况的测量实验。

在 1999-2011 年 5 月，公布了一个移动设备在一般情况下电池电量使用效率的一个综合测量结果[17]。作者公布并提供了各种数据，这些数据是关于移动设备的研究，建模和减少耗电量的因素的简单介绍。根据 Balasubramanian 在[18]中说明，我们讨论一种使用电量情况，在这种情况下，电量花费在下载上传 xKb 的数据流量，其包含三部分：ramp energy，transmission energy，tail energy。R(X)表示 ramp 和 transmission 花费 XB 的数据流量发送数据。Tail 每秒使用情况用 E 表示。在 WIFI 情况下，没有 ramp。在这种情况下，R(X)表示屏幕消耗的能量和 transfer 能量的和。在这种情况下 tail 花费的能量为 0。传输一个包花费的总能量取决于 interface 打开的时间。用 M 表示保持 interface 是打开状态的情况下的电量消耗情况，每秒所需花费的电量。最后，用 T 代表 tail。由于 KA 发送的周期比 tail-time 更长，如果没有任何数据需要传输，每个 KA 的电量的消耗就比 tail-time 高很多。这种情况下，通过减少 KA 的数量，我们可以减少总电量的消耗。

D.通过迭代探测来测量网络参数

在各论文著作中迭代探测已经被大量使用来测量不同网络参数，例如端对端的可用带宽(avail-bw)的可变大小，TCP 滑动窗口等。这个滑动窗口的大小是在慢开始情况下，每次收到确认都会扩大到原来发送数据的二倍的情况，在后来的增长模式下，每次迭代都会增加。任何时候检测到丢包了，慢开始阈值被设置为一半拥塞窗口的大小，全过程重启[19]。

Jain 迭代探测端到端的可用宽带大小。他们的测量方法的理论 Self-Loading Periodic Streams(SLoPS)，用一种被叫做 pathload[20]的工具实现了。其他可用来探测的技术包括 Bfind[21]，PTR[22]，TOPP[23]，pathChirp[24]等，Jain 使用了迭代方法已经测出了可用宽带的可变范围[25]。

第三章 动态提高 KA 间隔的技术

这部分中，我们描述我们所推荐的技术来提高 TCP 连接中 KA 间隔，所有的方案都使用迭代方式反复尽可能长间隔的发送 KA 包，一直到找到一个会令这个连接断掉的最大范围。这个范围我们视为 KA 的最佳间隔。

图二是了一个流程图，展示了如何找出这个最佳 KA 间隔。第一步打开一个独立的与目标服务器相连的 TCP 连接。如果这个连接在连接建立过程中就被关闭了，则连接可能已被破坏，并且可能是影响到了已经存在的其他连接。因此，我们在一个独立的连接中管理这个迭代，我们把这个连接看成测试连接。

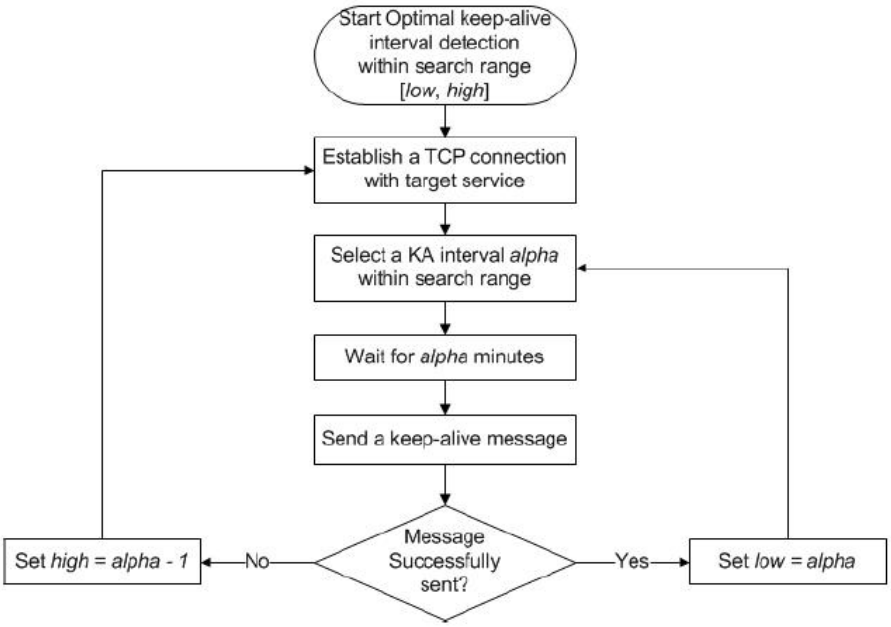


Fig. 2. Flow chart of KA interval detection technique.

随着测试提高间隔，新的间隔可以被告知连接已经关闭了。使用已知的最大的 KA 间隔。这个间隔也被用来作为查找最佳间隔的最低标准。我们也在这个搜索空间指定了一个更高的范围，更高的范围比已知不能工作的 KA 间隔少一分钟，注意到我们使用这个间隔便捷范围单位为分钟，在图中，查找范围使用[low,high]表示。

从更低的起始位置开始，我们尽力提高通过猜测新的 KA 间隔，一旦做出猜测，连接大多数都保持静默。后来，一个 KA 被发送来检查连接是否是连通的，如果连通，意味着我们猜测的 KA 间隔可以保证连接是连通的。随后，使用一个更长的 KA 间隔并进行测试，相反，如果连接断了，我们需要降低猜测并且再次测试，这个过程会一直持续下去，直到两个猜测之间间隔小于 1 分钟。

为了找到最佳 KA 间隔，我们使用了三种不同技术，他们之间的不同点在于他们怎样选择下一个 KA 间隔来测试。技术分别为二分查找，指数查找和复合查找。直觉上我们通常想到二分法来解决这个问题，这花费了最少的时间来查找最佳 KA 间隔。然而这个方法问题在于在数据传输中，KA 间隔不会提高的情况下，花费太长时间。由于这样的原因，我们随后测试了指数搜索方法。最初使用了更短的搜索时间，立刻改善了 KA 间隔在数据连接中，然而，指数查找在找出 KA 间隔时花费了大量时间找出最佳 KA 间隔，因此，我们最终组合这两种处理，我们称之为复合查找，它表现良好。

对于每一种技术，我们用 1 分钟初始化 low，因为它比家庭网络中最小 KA 间隔还小，我们在[1]部分已经给出了这个数据。指数和复合搜索技术不需要指定一个在搜索范围中指定一个更高的范围。换句话说，我们初始化最高为无限大，对于二分法搜索，我们用 128 分钟初始化最大值。这是一个二的指数次幂，也比 IETF 协议略高一点。

A.二分法搜索

在每一次循环中，这个查找范围的中间值被作为下次将要被测试的 KA 间隔的中间值。一次测试完成了，被测试的查找范围被分成两半。如果测试成功，我们查找这个第二半来初始化这个查找范围来找出一个更好的间隔。另一方面，如果查询失败，我们继续用前一半来初始化查找范围。这个过程一直持续下去。例如，如果一个 NAT 的超时时间是 24 分钟，查找范围为[1, 128]，这个间隔的测试顺序为:

$$ProbeSeq_{24}^{binary} = \{65^*, 33^*, 17, 25^*, 21, 23, 24\}$$

带‘*’的表示失败了

B.指数搜索

在这个方法中，与二分法不同的是这里使用了 2 的幂次方增长。因此，前几个增长的间隔是 2,4,8,16,32 分钟等等。如果下一个将要被测试的间隔大于 high，那么没有必要测试测试这个间隔。代替，low 将增长到最后的测试的间隔；并且这个间隔的的不同以一重启。另一方面，如果下一个将要测试的间隔超时了，连接丢失了。在这种情况下，除了修改上面提到的，high 被减到小于失败的测试的间隔减一。一直这样下去，直到最佳 KA 被找到。例如，如果最佳间隔是 24 分钟，间隔的测试的序列为

$$ProbeSeq_{24}^{exp} = \{2, 4, 8, 16, 32^*, 17, 19, 23, 31^*, 24, 26^*, 25^*\}$$

带‘*’的表示失败了

C.综合查找

综合查找是一个二分查找和指数查找的综合。原来，它表现的就想指数查找一样。与二分法不同的是这里使用了 2 的幂次方增长。(这里是不是错了)。当一个被测试的间隔在实际中超时了，这个测试连接将会终止。这样的话，low 增加到最后一次测试成功的间隔，high 减小到比不成功间隔减小一，后来，用二分查找在新的查询范围内。例如，如果超时间隔为 24 分钟，间隔的测试序列为：

$$ProbeSeq_{24}^{comp} = \{2, 4, 8, 16, 32^*, 24, 28^*, 26^*, 25^*\}$$

带‘*’的表示失败了。

第四章 KA 方案的分割范围

我们分析出了两个评测指标。即在找出最佳 KA 间隔的测试次数(Probe Count)，以及做花费的总时间(Convergence Time)。我们的方案基于这个两个指标进行比较。

用阿尔法(α)表示最佳间隔。设置搜索范围为[1, h]，对于二分法，h=2^k，k 取值大于等于 1。对于指数和复合搜索，h =∞，使用 N(α)表示检测最佳间隔所需要探测的次数。分别用 N(best)，N(worst)，N(avg)表示最好，最坏，平均值。用 T(α)表示查找最佳间隔所需要的时间。这被叫做收敛时间。分别用 T(best)，T(worst)，T(avg)表示最佳，最坏，平均收敛时间。下面，对于不同的检测技术，我们获得了这些属性的取值范围。由于篇幅限制，我们只列举了二分法搜索的推导过程，对于其他的推导过程，我们这里只记录了结果。读者可以参考[26]查看详细信息。

A.测试次数

二分法中：

$$N_{best}^{binary} = N_{avg}^{binary} = N_{worst}^{binary} = \lg h \qquad (1)$$

指数搜索中

$$N_{best}^{exp} = 1 \qquad (2)$$

$$N_{worst}^{exp} = \begin{cases} \lg h + 2 & \text{when } h \le 4 \\ \frac{\lg h(\lg h+1)}{2} & \text{otherwise.} \end{cases} \qquad (3)$$

$$N_{avg}^{exp} = \frac{\lg h(1+\lg h)}{4} + 1 + \frac{1}{h} \qquad (4)$$

在复合搜索中

$$N_{best}^{comp} = 1 \qquad (5)$$

$$N_{worst}^{comp} = 1 + 2\lg h \qquad (6)$$

$$N_{avg}^{comp} = 2\lg h + \frac{2}{h}(2+\lg h) - 3 \qquad (7)$$

不同测试方法中的测试次数都在表 1 中被列举出来了

TABLE I
PROBE COUNT COMPARISON AMONG TECHNIQUES TO DYNAMICALLY IMPROVE KA INTERVAL.

	Best	Average	Worst
Binary search	$O(\lg h)$	$O(\lg h)$	$O(\lg h)$
Exponential search	1	$O(\lg^2 h)$	$O(\lg^2 h)$
Composite search	1	$O(\lg h)$	$O(\lg h)$

B.收敛时间

在二分搜索中，第一个测试花费 1+2^(k-1)单位时间。如果 α >=1+2^(k-1)，第二次测试需要在第一次完成测试后花费 1+2^(k-1)+2^(k-2)个单位时间。另一方面，如果 α <=2^(k-1)，那么第二次测试仅仅需要花费 1+2^(k-2)个单位时间。用 α k-1 α k-2⋯ α 1 α 0 表示 α -1 的二进制。因此，我们可以写出如下表达式：

$$T^{binary}(\alpha) = (h + \lg h - 1) + \sum_{i=1}^{\lg h - 1} a_i i 2^i \qquad (8)$$

最好情况下是当 α 等于 1 或 2 时，这时这个值为 2^k+k-1

$$T^{binary}_{best} = h + \lg h - 1 \qquad (9)$$

最坏的情况发生在当 α =2^k 或 α =2^k-1

$$\begin{aligned} T^{binary}_{worst} &= (2^k + k - 1) + \sum_{i=1}^{k-1} i 2^i \\ &= h \lg h + \lg h - h + 1 \end{aligned} \qquad (10)$$

平均收敛时间为

$$\begin{aligned} T^{binary}_{avg} &= \frac{1}{2^k} \sum_{\alpha=1}^{2^k} T(\alpha) \\ &= (\frac{h}{2} + 1) \lg h \end{aligned} \qquad (11)$$

对于指数搜索，我们有

$$T^{exp}_{best} = 2 \qquad (12)$$

$$T_{worst}^{exp} = \begin{cases} 5h-1 & \text{when } h \leq 8 \\ \frac{\lg^2 h - 3lgh + 12}{2} h - \frac{\lg^3 h + 11 \lg h + 36}{6} & \text{otherwise.} \end{cases} \tag{13}$$

$$T_{avg}^{exp} = \frac{\lg^2 h - 3\lg h + 24}{8} h - \frac{\lg^3 h + 23 \lg h - 24}{24} \tag{14}$$

对于复合搜索我们有

$$T_{best}^{comp} = 2 \tag{15}$$

$$T_{worst}^{comp} = h \lg h + 5h - 3 \tag{16}$$

$$T_{avg}^{comp} = (\frac{h}{2} + 1) \lg h + \frac{2}{3} h + 3 - \frac{5}{3h} \tag{17}$$

TABLE II
CONVERGENCE TIME COMPARISON AMONG TECHNIQUES TO DYNAMICALLY IMPROVE KA INTERVAL.

	Best	**Average**	**Worst**
Binary search	$O(h)$	$O(h \lg h)$	$O(h \lg h)$
Exponential search	2	$O(h \lg^2 h)$	$O(h \lg^2 h)$
Composite search	2	$O(h \lg h)$	$O(h \lg h)$

第五章 实验

我们已经实现了在不同 Omnet++仿真平台上检测最佳间隔的技术[27]。我们做了一个实验，客服端通过一个单一的中间件连接到服务器。尽管在实际中，一个连接可能穿过一系列的不同超时间隔的 NAT 和防火墙，但是在这个路径中，其中的最小的超时间隔仍然是适用的。在这个中间，一系列的中间件可以被更小超时间隔的中间件代替。在图三中，展示了仿真设置的拓扑结构。

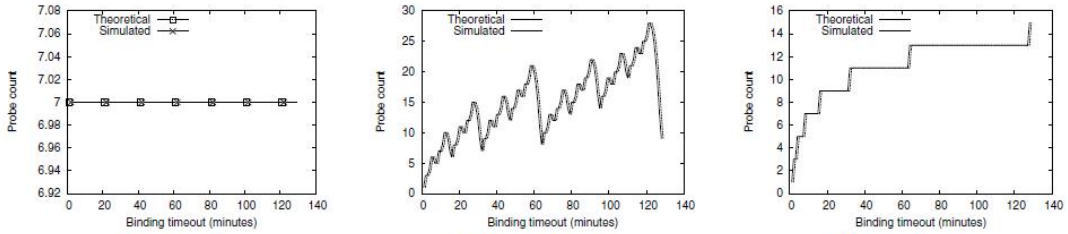
Fig. 3. Simulation topology.

从一个节点到中间件的延时被设置为 10ms，中间件和服务端的延时被设置为 100ms。这个选择是基于分别对网关往返时延以及云服务的观测得出的，从不同的接入点接入到 CSE 或 BUET。只要这个时延小于一分钟，我们的实验结果就是有效的。

A.分析范围的正确性

通过模拟，在图四 a 中绘出了二分法搜索的测试次数。在同一张图上也画出了理论曲线，可以看出，它们是相同的。这是被期望看到的，这是因为在给定范围内测试次数是不与任何具体的超时时间相关。

图 4b 和图 4c 分别绘制了与超时时间相关的指数搜索和复合搜索的测试次数的拟合函数。在这两图中，可以看出实验曲线和理论曲线有一分钟的偏差。也就是说，对于任何观察到的测试次数 α 都与理论上的测试次数 α -1 相等，这是因为有网络时延 T。由于中间件在 α 单位时间后会连接超时，如果一个节点在 α 单位时间的沉默后发送 KA 数据包，这个 KA 包会在中间件 α +t 单位时间后收到这个 KA 包，其中 0< τ « α，因此这个连接断掉了。因此，从节点角度看，连接超时时间为 α -1。由于这个原因，实验的收敛时间曲线(简洁起见，未画出来)也在 Y 轴方向偏离理论值一个单位。



(a) Binary search. (b) Exponential search. (c) Composite search.

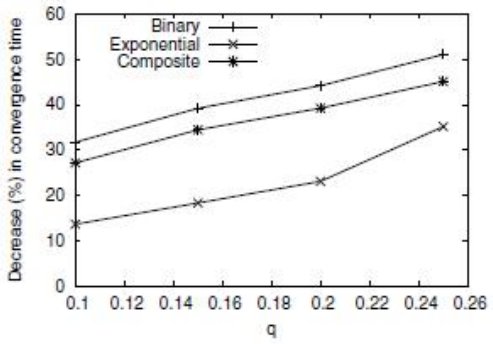
Fig. 4. Probe count of different search techniques in presence of network delay.

B.测试 KA 间隔花费的时间

迭代探测技术的长收敛时间可能导致服务器的增加负载。为了缓解服务器压力，我们可以牺牲探测 KA 间隔的准确性来减少收敛时间。在我们的算法中，我们介绍调节参数 q 来达到这样的目的。使得 a 是最佳存活间隔。为了努力减少收敛时间，让我们找出次最佳的 KA 间隔 a。我们想要允许一个误差，不超过 a 的 q 分之一。有 a 丿>=(1-q)a，只要(high-low)/high>q 我们就一直测试下去。在这种情况下，当测试完成时，我们可以写出：

$$\begin{aligned}\alpha' &= low \\ &\geq (1-q)high \\ &\geq (1-q)\alpha\end{aligned}$$

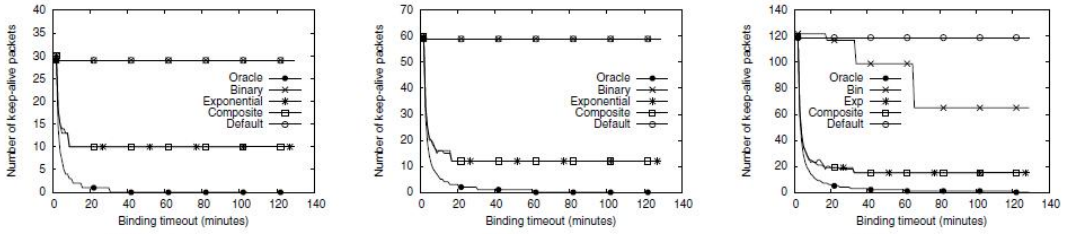
简单的修改后，我们可以找到一个次最佳的 KA 间隔，这个间隔与实际的最佳间隔之间的误差小与实际 KA 间隔的 q 分之一。我们在我们的算法中对于不同的 q 的值实施了这一个改变来找出对于收敛时间的影响。正如我们在图五中看出的，收敛时间减少的平均值与 q 具有线性关系。二分搜索减少的最大。我们没有必要把 q 的值硬编码到算法中。相反，它可以动态调整。服务器可以在不同的时间对不同的节点设置不同的 q 的值。这个值可以被插入到由客户端发送的 KA 消息中。



C.发送 KA 包的数量

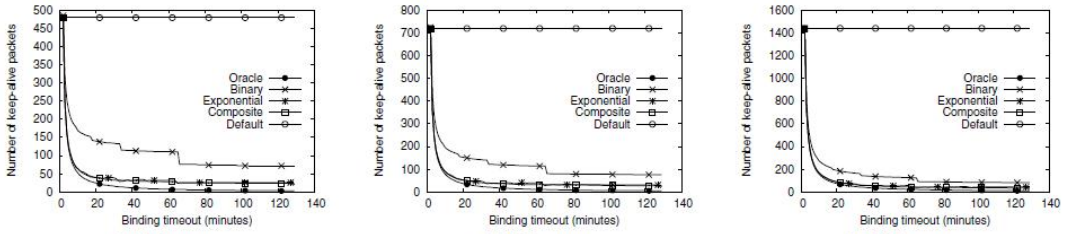
从测试开始，我们计数一段时间内，我们通过连接发送 KA 包的总数。这个数量越小越好。这是因为每次发送数据包，设备都会处于高功率状态，如果设备闲置就不会这样。我们做这个实验来反映一些真实的场景，设备在一段时间内在同样网络设置的环境中。

首先，我们分别进行了 30 分钟和一个小时。这两个在任何会议组织中都被认为是典型时间。我们也在 3 小时情况下进行了实验。这与研讨课，讨论会，或研讨会的时间相同。我们标出了发送 KA 包的数量和对于中间件测试连接的数量。所得曲线如图六



(a) 30 minute run. (b) 1 hour run. (c) 2 hour run.

Fig. 6. Number of keep-alive packets sent during meeting or seminar scenario.



(a) 8 hour run. (b) 12 hour run. (c) 24 hour run.

Fig. 7. Number of keep-alive packets sent during longer durations.

被标记为默认的的曲线是在没有测试的情况下画出的曲线。在这种情况下，每分钟发送一个 KA 包。另一方面，标记为甲骨文(Oracle)的曲线是在已知最佳间隔情况下的系统的行为曲线。很显然，在图 6a 和图 6b 中可以看出，对于二分查找这个曲线与默认曲线相匹配。这是因为等待发送第一测试帧的时间比场景的持续时间还要长。这样，二分搜索技术就没办法提高。图 6c 显示了使用二分搜索法发送 KA 的数量。在运行时允许一种或多种测试方法，可以提高 KA 间隔。

相反，让我们看一下指数搜索和复合搜索的表现。如图六所示。这些方法的曲线看起来相同。这两种技术都可以显著减少发送保活包的数量。这是因为在这些方法中，测试的时候 KA 间隔的改善，在开始的时候提高的更容易。在图 6c 中画出了二十单位时间内的 Oracle 曲线。

接下来，我们进行了 8 小时，12 小时，24 小时的测试。在这些情况下的 KA 包发送的数量，不同的测试结果显示在图 7 中。在所有的实验中，复合搜索和指数搜索几乎是相同的，并且他们随着时间越来越接近 Oracle 曲线。

对于不同的超时间隔，我们计算相比于没有进行测试的情况下，不同搜索技术 KA 包发

送数量减少的百分比。然后我们测试在[1, 128]范围内，对于整个的超时范围的取平均值。在表三中，列出了 KA 发送数据包的平均减少的数量。基于该结果，可以很明显看出相比较指数搜索和复合搜索，更应该选择二分搜索。并且由于复合搜索另外两种的优点，它是最好的选择。

TABLE III
AVERAGE REDUCTION (%) OF NUMBER OF KEEP-ALIVES SENT WHEN TESTING IS PERFORMED TO IMPROVE THE KEEP-ALIVE INTERVAL FROM THE CONSERVATIVE DEFAULT VALUE.

Experiment duration	Binary search	Exponential search	Composite search
30 minutes	0	64.30	64.25
1 hour	0	77.79	77.83
2 hours	26.79	84.92	85.05
6 hours	71.62	90.88	91.06
8 hours	77.52	91.74	92.09
12 hours	83.47	92.84	93.13
24 hours	89.61	94.12	94.39

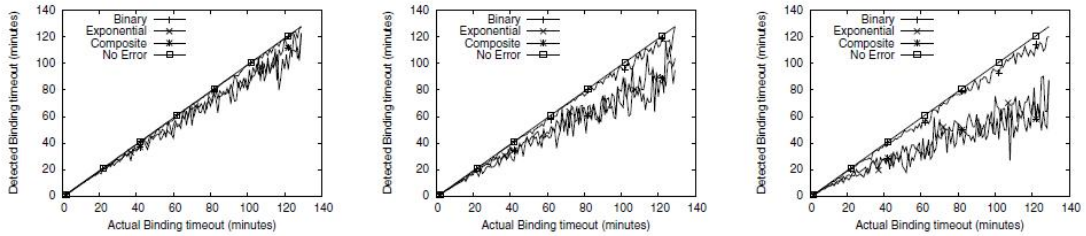
D.发送包失败的影响

到现在为止，我们一直假设，在测试连接中，KA 包发送失败是由于网络中间件太长时间没有活动二丢弃这个连接。然而，一个数据包被丢弃也可能是由于网络的一些瞬态问题。发送者没办法区分失败的原因，因此，我们的技术也可能并没有改善 KA 的间隔。

参数在这个实验中，我们尽力试图模拟瞬时网络故障观察不同搜索技术的搜索行为。为了模拟这个故障，我们定义了一个参数 p，p 表示 KA 包可能失败的概率。消息将失败是相互独立的。对于不同的参数 p，我们观察不同的被检测到的超时间隔的值。对于每个 p 的值，我们重复 20 次试验，对结果取平均值。图 8 分别绘制出了检测到的超时间隔和实际的超时间隔。被标记为“没有错误”的曲线表示检测过程中没有涉及错误。从曲线中可以很明显看出，包失败的影响不同忽视。

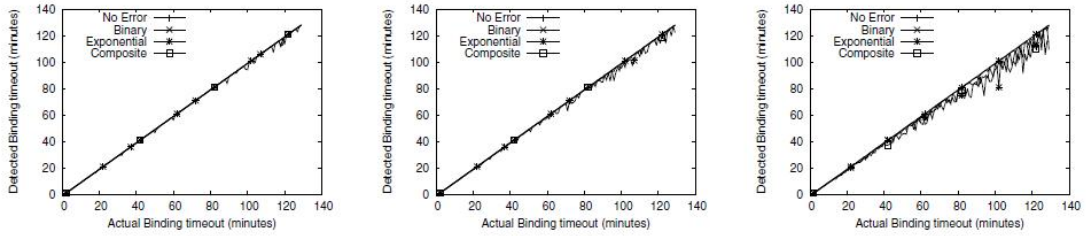
E.包失败重试

我们如下修改没个算法：当 KA 消息发送失败时，我们重新建立 TCP 连接，再次测试相同的时间间隔。不会两个包都遇到了瞬时故障，如果后一次尝试也失败了，我们认为已经超过了最佳 KA 间隔。经过这样的修改，我们重新进行试验。在图九中展示结果。对于 p=0.02 和 p=0.05，检测到的超时间隔中错误几乎是可以忽略不计的。



(a) $p = 0.02$. (b) $p = 0.05$. (c) $p = 0.10$.

Fig. 8. Detected vs. actual binding timeout in presence of transient network errors.



(a) $p = 0.02$. (b) $p = 0.05$. (c) $p = 0.10$.

Fig. 9. Detected vs. actual binding timeout when retry scheme is applied in presence of transient network errors.

对于 p=0.01，二分查找对于应对这样的暂时性网络问题是相当成功的。平均误差小于 0.5%。对于指数搜索，测试超时间隔的平均误差在 5%左右，对于复合搜索，误差大约为 3.5%。虽然误差不可以忽略不计，但是我们认为这是可以容忍的。

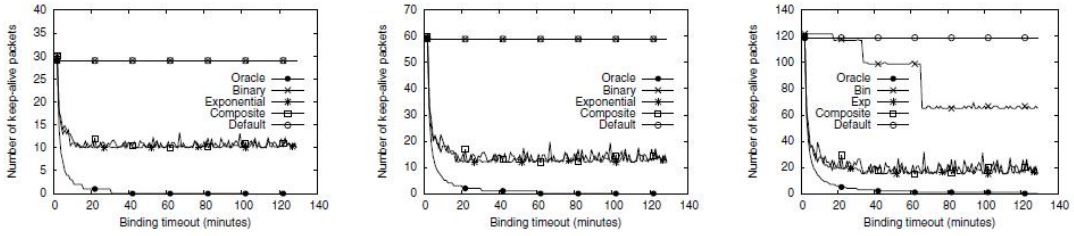
对于单纯的重试方法，对于每个有效的 KA 时间失败时，都需要花费二倍的时间。因此，收敛时间显著增加。二分搜索的收敛时间最长，越站到平均水平的 60%。由于重试的影响，每个测试技术的测试次数也增加了。既然测试超时间隔的错误在可容忍范围内，并且重试对于收敛时间和测试次数的影响太大了，我们将不会进行简单的重试。

对于重试方案，我们想知道在不同技术情况下在减少发送过来的数据和测试连接情况下，保活包的数量怎样变化。与前面一样，我们在真实世界场景下，我们进行不同事件的测试。该包失败概率 p 被设置为 0.02，q 被设定为 0.10。(回想一下，q 是在检测到超时时允许的误差，实际超时的百分比)

对每一个超时间隔是二十次重复试验的平均值。在图十和图十一中给出了结果。这些曲线与图六和图七很类似。在表四种列出了 KA 包平均减少的百分比。在表三中可以看到相应的值。

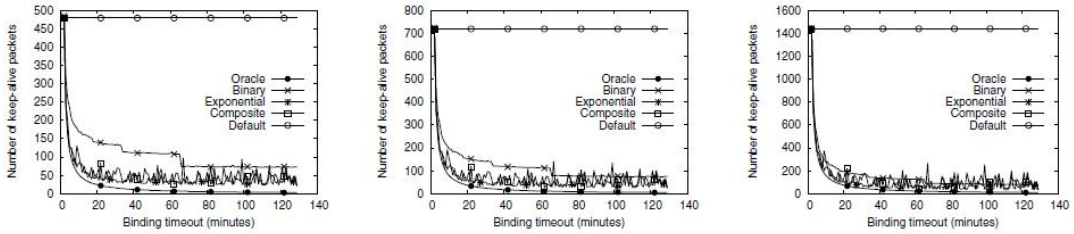
TABLE IV
AVERAGE REDUCTION (%) OF NUMBER OF KEEP-ALIVES SENT WHEN TESTING IS PERFORMED TO IMPROVE THE KEEP-ALIVE INTERVAL FROM THE CONSERVATIVE DEFAULT VALUE. HERE $p = 0.02$ AND $q = 0.10$.

Experiment duration	Binary search	Exponential search	Composite search
30 minutes	0	62.37	62.37
1 hour	0	75.44	75.58
2 hours	26.45	82.39	82.62
6 hours	71.59	88.19	88.56
8 hours	77.57	89.03	89.52
12 hours	83.59	90.10	90.56
24 hours	89.61	91.32	91.69



(a) 30 minute run. (b) 1 hour run. (c) 2 hour run.

Fig. 10. Number of keep-alive packets sent during meeting or seminar scenario. ($p = 0.02$, $q = 0.10$).



(a) 8 hour run. (b) 12 hour run. (c) 24 hour run.

Fig. 11. Number of keep-alive packets sent during longer durations. ($p = 0.02$, $q = 0.10$).

因此，我们可以得出这样的结论：重试可以很好的应对短暂的网络故障，并能够很大程度减少 KA 包和测试连接的包的数量。使用 q 参数，检测过程中牺牲了一些准确性，减少了收敛时间。即使这样，KA 包数量的减少与以前的实验结果也是很匹配的。总体而言，基于所有的这些实验，我们认为，复合搜索技术应该被用来动态改善 KA 间隔。

F.实现的概念证明

我们在 Android 设备上实现了没有重试的复合搜索技术。我们在 kopottakha.cs.uiuc.edu 上 8080 端口上部署了服务器，让设备连接到它之上。这个概念证明系统仅仅使用了一个测试连接到最佳 KA 间隔。在数据连接中没有修改 KA 间隔——这需要在操作系统中更改代码。我们在两个不同的移动运营商使用我们的系统。检测出的最佳 KA 间隔是 9 分钟和 10 分钟。这个真正实施的测试序列和收敛时间与我们的分析范围相匹配。我们也在 WIFI 网络中进行了测试。

第六章 结论

在这项研究中，我们采用多次迭代探测技术动态适应 TCP 长连接时间间隔。包括二分搜索、指数搜索和复合搜索。我们在仿真平台上进行理论分析，同时进行试验比较这些技术。据我们所知，以前没有人做过这些工作。我们通过改变不同的参数评估我们的技术性能，发现复合搜索是最好的选择。

其他的搜索技术可以在以后的研究工作中进行讨论。特别的，多个测试连接搜索可以显著减少收敛时间，并且迅速提高 KA 间隔。然而，注意不要使用太多的连接造成超载。偶尔情况下，由于网络基础设施的改变，中间件的超时时间可能减少。这种情况下，数据连接将经历频繁的断线。应该进行试验制定策略，衬托这个不稳定的状态(频繁断线)。最后，可以编写可以插入疯了设备的提高 KA 间隔的库。这样的库的 API 和协议还有待进一步研究。

致谢

作者要感谢所有给出宝贵意见的匿名审稿人。

参考文献

[1] S. H ¨ at ¨ onen, A. Nyrhinen, L. Eggert, S. Strowes, P. Sarolahti, and M. Kojo, “An experimental study of home gateway characteristics,” in Proceedings of the 10th ACM SIGCOMM conference on Internet measurement. ACM, 2010, pp. 260–266.

[2] R. Braden, “Requirements for Internet hosts-communication layers,” October 1989, RFC 1122 (Standard).

[3] S. Guha, K. Biswas, B. Ford, S. Sivakumar, and P. Srisuresh, “NAT Behavioral requirements for TCP,” October 2008, RFC 5382 (Best Current Practice).

[4] M. Stiemerling, E. Davies, C. Aoun, and H. Tschofenig, “NAT/Firewall NSIS Signaling Layer Protocol (NSLP),” October 2010, RFC 5973 (Experimental).

[5] M. Stiemerling, J. Quittek, and C. Cadar, “NEC’s Simple Middlebox Configuration (SIMCO) Protocol Version 3.0,” May 2006, RFC 4540 (Experimental).

[6] “Understanding Direct Push,” http://technet.microsoft.com/en-us/library/aa997252(EXCHG.80).aspx, [Online; Last accessed on 22-Oct-2015].

[7] “Heartbeat Interval Adjustment,” http://technet.microsoft.com/en-us/library/cc182270.aspx, [Online; Last accessed on 22-Oct-2015].

[8] “Apple Push Notification Service,” https://developer.apple.com/library/ios/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/Chapters/ApplePushService.html, [Online;Last accessed on 22-Oct-2015].

[9] “Google Cloud Messaging: Overview,” https://developers.google.com/cloud-messaging/gcm, [Online; Last accessed on 22-Oct-2015].

[10] “Push Notifications (Windows Phone),” https://msdn.microsoft.com/en-us/library/hh221549.aspx, [Online; Last accessed on 22-Oct-2015].

[11] “Windows Push Notification Services (WNS) overview (WindowsRuntime apps),” http://msdn.microsoft.com/en-us/library/windows/apps/hh913756.aspx, [Online; Last accessed on 22-Oct-2015].

[12] S. R. Gatta, K. Srinivasan, O. N. Ertugay, D. G. Thaler, D. A. Anipko,J. Vanturennout, M. S. Rahman, and P. R. Gaddehosur, “Keep alive management,” November 2014, US Patent No. 8,892,710 B2.

[13] S. Herzog, R. Qureshi, J. Raastroem, X. Bao, R. Bansal, Q. Zhang, and S. M. Bragg, “Determining an efficient keep-alive interval for a network connection,” February 2013, US Patent No. 8,375,134 B2.

[14] R. Seggelmann, M. Tuexen, and M. Williams, “Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS) Heartbeat Extension,” February 2012, RFC 6520 (Standard).

[15] T. Dierks and E. Rescorla, “The Transport Layer Security (TLS) protocol version 1.2,” August 2008, RFC 5246 (Standard).

[16] H. Haverinen, J. Siren, and P. Eronen, “Energy consumption of alwayson applications in

WCDMA networks," in Proceedings of IEEE Vehicular Technology Conference. IEEE, April 2007, pp. 964–968.
[17] N. Vallina-Rodriguez and J. Crowcroft, "Energy management techniques in modern mobile handsets," Communications Surveys & Tutorials,
IEEE, vol. 15, no. 1, pp. 179–198, 2013.
[18] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani, "Energy consumption in mobile phones: a measurement study and implications
for network applications," in Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference. ACM, 2009, pp. 280–293.
[19] D. J. Wetherall and A. S. Tanenbaum, "Computer Networks," 1996.
[20] M. Jain and C. Dovrolis, "End-to-end available bandwidth: Measurement methodology, dynamics, and relation with TCP throughput," in ACM SIGCOMM Computer Communication Review, vol. 32, no. 4. ACM, 2002, pp. 295–308.
[21] A. Akella, S. Seshan, and A. Shaikh, "An empirical evaluation of widearea internet bottlenecks," in Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement. ACM, 2003, pp. 101–114.
[22] N. Hu and P. Steenkiste, "Evaluation and characterization of available bandwidth probing techniques," IEEE Journal on Selected Areas in Communications, vol. 21, no. 6, pp. 879–894, 2003.
[23] B. Melander, M. Bjorkman, and P. Gunningberg, "A new end-to-end probing and analysis method for estimating bandwidth bottlenecks," in Global Telecommunications Conference, 2000. GLOBECOM'00, vol. 1. IEEE, 2000, pp. 415–420.
[24] V. Ribeiro, R. Riedi, R. Baraniuk, J. Navratil, and L. Cottrell, "pathChirp: Efficient available bandwidth estimation for network paths," in Proceedings of Passive and active measurement (PAM) workshop, April 2003.
[25] M. Jain and C. Dovrolis, "End-to-end estimation of the available bandwidth variation range," in ACM SIGMETRICS Performance Evaluation Review, vol. 33, no. 1. ACM, 2005, pp. 265–276.
[26] M. S. Rahman, "On Approaches to Detect Optimal Keep-alive Interval of TCP Connections." http://1drv.ms/1kktYFu, M.Sc. thesis, submitted to the Department of Computer Science and Engineering (CSE), Bangladesh University of Engineering and Technology (BUET). [Online; Last accessed on 22-Oct-2015].
[27] "Omnet++," http://www.omnetpp.org/, [Online; Last accessed on 22-Oct-2015].