浙江大學

本科生毕业论文
开题报告

学生姓名：瞿泽宁

学生学号：3080102955

指导教师：彭群生

专　　业：2008 级 计算机科学与技术

学　　院：计算机科学与技术学院

一、题目：_____

二、指导教师对开题报告、外文翻译和文献综述的具体要求：

指导教师(签名)：

年　　月　　日

毕业论文开题报告、外文翻译和文献综述考核

导师对开题报告、外文翻译和文献综述评语及成绩评定：

成绩比例	开题报告占 (20%)	中期报告占 (10%)	外文翻译占 (10%)
分值			

导师签字 _____

年　月　日

答辩小组对开题报告、外文翻译和文献综述评语及成绩评定：

成绩比例	开题报告占 (20%)	文献综述占 (10%)	外文翻译占 (10%)
分值			

答辩小组负责人(签名) _____

年　月　日

目 录

第 1 章　本科毕业论文开题报告

1.1　课题背景

1.2　目标和内容

1.3　可行性分析

1.4　研究方案和关键技术考虑

1.5　预期研究结果

1.6　进度计划

第 2 章　本科毕业论文文献综述

第 3 章　本科毕业论文外文翻译

摘要

当今世界,网络已被内容(content)的散布和获取所主导,而网络技术却仍然只言主机间连接而不及其它。为获取网络内容和服务,须要建立起一个从网络用户所关心的(内容)到网络中的位置的映射。我们提出内容为心网络(Content-Centric Networking 简称 CCN)。它以内容为原语——把网络位置与网络身份、安全和访问分离开来,用名字访问内容。我们从 IP 推演出适用于命名内容的路由算法。使用这些算法,我们可以使网络的可扩展性、安全性和性能同时达到要求。我们实现了我们的体系结构的基本框架,用安全文件下载和 VoIP 电话两个例子演示了网络的性能和可恢复性。

类别和学科描述符

C.2.1 [计算机系统组织]:网络结构与设计; C.2.2 [计算机系统组织]:网络协议

通用术语

设计,实验,性能,安全

3.1 引言

今天的互联网背后的工程原理和体系结构创建于二十世纪六、七十年代。网络致力于解决的问题是资源共享的问题——远程使用如读卡器、高速磁带机甚至超级计算机等珍贵资源。这样产生的通信模型是一个仅仅存在于两台机器之间的对话模型,一台机器希望使用资源,另一台机器为前者提供资源。因此 IP 包含有两个地址,一个是发送方的,一个是目的地机器的,并且互联网上几乎所有的通信都包含 TCP 对话。

自包交换网络被创造的 50 年来,计算机及其附件已经成为廉价、普适的商品。互联网的连通性和低价的存储成本使得对海量新内容的访问成为可能——仅 2008 年就有 500 艾字节的信息被生产出来 [13]。人们在意的是互联网所承载的东西,而通信仍然在谈论这些东西都在哪些地方。

我们看到这个矛盾给用户带来三方面的(负面)影响。

可用性 为快速、可靠地访问内容,不仅须要如 CDN 和 P2P 网络这样繁冗的,预先设计好的,面向应用软件的机制,而且还会大幅增加带宽成本。

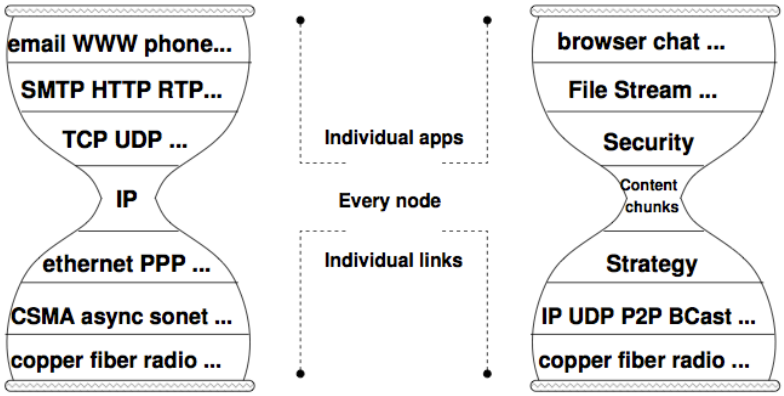

图 3.1 CCN 把网络协议栈中全局性的成分由 IP 换为了命名内容块

安全性 对内容的信任这一概念被偷换为不那么可信的对物理位置和网络连接的的信任。

位置迁就 建立从内容到物理位置的映射使网络服务的配置和实现工作都变得复杂。

解决这些问题的直接办法就是把网络中的“在哪里”(where)替换为“是什么”(what)。主机到主机对话是为了解决六十年代的问题而提出的抽象模型。对于如今的通信问题,我们认为命名数据(named data)是比命名主机(named hosts)更好的模型。我们提出内容为心网络(CCN),一个建立于命名数据之上的网络架构。即使是在 CCN 的最底层,我们也没有位置的概念——一个 CCN 包的“地址”是一个内容的名字,而不是这个内容的位置。不过,我们也保留了当年使 TCP/IP 简单、稳健、可扩展的那些设计决策。

图 3.1 比较了 IP 和 CCN 协议栈。其中很多层都反映了双端约定;例如,第二层的成帧协议就反映了物理层上两端的约定,而第四层的传输协议就是一些信息生产者和消费者之间的约定。唯一一个需要全局约定的层就是第三层,即网络层。 IP 的成功在很大程度上要归功于它的网络层的简洁和它在网络第二层处提出的低要求,即:无状态,不可靠,无序,尽力交付。 CCN 的网络层(详见 3.3)与 IP 的网络层很相似且 CCN 对第二层的要求更少,这使得它非常具有吸引力。除此之外,CCN 可以单独成层,放置于任何一层之上,包括放置于 IP 层之上。

CCN 与 IP 有几点关键性的不同。其中两点即策略和安全已经以新层的方式在协议栈中展示了出来。由于 CCN 与第二层的关系更为简单,它可以最大限度地利用众多同时存在的网络的连通性(如以太网,3G ,蓝牙和 802.11 等等)。 CCN 的策略层(详见 3.3.3)为在不断变化的环境下最好地开发利用多种网络的连通性不断地做出细粒度、动态优化的选择。 CCN 确保的是内容本身的安全(详见 3.5),而不是内容走过的连接的安全。因此 CCN 没有许多基于主机网络的弱点,而不像 IP 网络那样深受其害。

我们在 3.2 至 3.5 中叙述了 CCN 的体系结构和运行方式。在 3.6 中

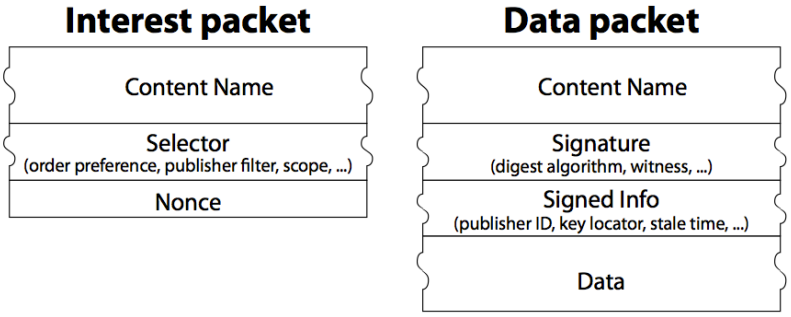

图 3.2 CCN 包的类型

我们评估了我们样板网络的性能。最后,在 3.7 和 3.8 中,我们讨论了相关工作并做出了结论。

3.2 CCN 节点模型

CCN 通信是由数据消费者驱动的。CCN 包分为两类:兴趣包(Interest)和数据包(Data)(如图 3.2)。消费者通过向所有连通之处广播自己的兴趣包来索要内容。任何一个收到兴趣包并且拥有满足该兴趣包的数据的节点都可以用一个数据包来回应该兴趣包。数据包仅为回应一个兴趣包而生,并且数据包产生之后这个收到的兴趣包就被“消费”了。[1] 因为兴趣包和数据包都用名字(name)来辨识内容,所以对同一个

[1]数据包和兴趣包因此是一对一的。它们保持这严格的流平衡。TCP 中有一个类似的流平衡,即数据包和 ack 包之间的平衡。这一平衡赋予了 TCP 可扩展性和强适应性 [20]。然而与 TCP 不同的是, CCN 模型对于多对多多点递送同样奏效(详见 3.3.1)。

内容感兴趣的多个节点可以共享一个广播介质。其间只要使用标准的多播风暴抑制技术即可控制网络负荷 [3]。

如果兴趣包中的内容名字(ContentName)是数据包中内容名字的前缀,那么我们称数据包“满足”了兴趣包。 CCN 名字是由若干(显式确定数量的)名字部件(component)组成的不透明二元对象(如图 3.4)。名字一般是结构化的,因此如果前缀匹配则意味着数据包的名字是兴趣包的名字的子树(详见 3.3.2)。 IP 使用了这种习惯来解析 < 网络,子网,主机 > 这一层次结构。经验表明这样做既可以有效地将路由和转发状态结构化地集成,又可以支持快速查找。[2] 这中匹配方式暗藏着这样一种可能性:内容尚未被生产出来,兴趣包就已经收到了——这使得发布者可以根据收到的数据请求实时地生产内容以回应兴趣。今天的网络中既有很多静态缓存着的数据,又有很多动态生成的数据, CCN 灵活变化的名字机制使得它不论应对哪种数据都游刃有余。 CCN 名字还可以是上下文

[2]虽然 CCN 的名字是不定长的并且通常比 IP 地址要长,但是它们可以被高效地检索。一个 IP 地址的结构不是显式的,而是隐式地由节点的转发表决定的。因此,对 IP 查找使用现代 O(1) 哈希技术是十分困难的。所以 IP 查找通常使用 log(n) 的基数树检索或者并行但昂贵的 TCAMs 高端硬件。由于 CCN 的名字结构是显示的,内容名字可以很容易地被哈希,以便查找。

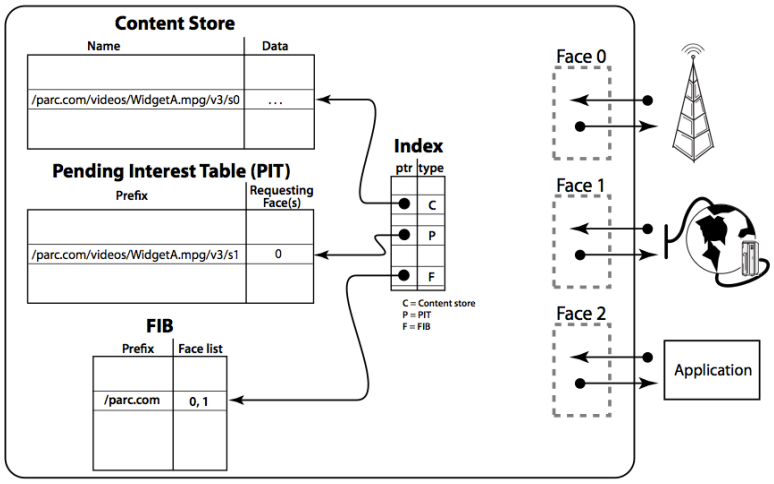

图 3.3 CCN 转发引擎模型

相关的,比如用 /ThisRoom/projector 与当前房间的投影仪交换信息,又如用 /Local/Friends 与本地(广播)环境上的任一好友交流。[3]

CCN 节点的基本运营方式与 IP 节点非常相似:当一个包到达一个 face 时,(节点)会先做一个最长名字匹配,后续操作由匹配结果而定。[4] 图 3.3 展示了 CCN 的转发引擎。它有三个主要的数据结构: FIB (Forwarding Information Base,转发中心),CS(Content Store,内容缓存)和 PIT(Pending Interest Table,待理兴趣表)。

FIB 用于将兴趣包转发给潜在数据源。它与 IP FIB 几乎相同,只是 CCN 的 FIB 会为每个兴趣包维护一个出口的列表,而 IP 的 FIB 仅需

[3]这第二个例子会用到 CCN 签名机制产生的身份信息使好友们能用固定的名字约会而不是使用复杂的枚举或是探测算法。换言之,名字表示谈话的内容,而签名表示谈话者在上下文中的身份,如“本地的一个好友”。

[4]这里原文中使用了英文单词 face 而非 interface,因为包不仅仅是在硬件接口之间被转发,也被路由器内的应用程序处理(详见 3.6)。

要为每个收到的包维护一个转发方向。这也反映了 CCN 在转发这方面并无限制这一事实。它允许多个数据源的存在,并允许并发地请求数据。

CS 与 IP 路由器的缓存大同小异,不过它们的置换策略略有不同。由于每个 IP 包仅仅属于一个点到点对话,它被转发了以后就不再具有任何意义了。因此 IP 路由器的做法是直接抛弃已转发的包并立即循环使用自己的缓存(MRU 置换)。 CCN 包是幂等的,能够自我鉴别和自我认证,因此每一个 CCN 包都有被很多用户重用的潜力(如许多主机阅读同一份电子报纸或许多用户观看同一段 Youtube 视频)。为了使多个用户分享同一内容的概率最大化(这会带来上行带宽需求最小化和下行延时最小化等诸多裨益), CCN 会尽可能多地储存到来的数据包(LRU 或 LFU 置换)。

PIT 记录了数据包向数据源的转发记录,这样当数据包回到路由器时就可以按照 PIT 中的记录被转回至数据申请者(们)。在 CCN 中,只有数据包会被路由,并在它们走向潜在数据源的路上沿途撒下“面包屑”,以便匹配的数据将来沿着这些面包屑逆行,直到找到数据申请者(们)。每一个 PIT 表项就是这样一片面包屑。当一个 PIT 表项被用于转发数据包一次之后(即数据包“消费”了对应兴趣包),这个 PIT 表项就应该被

抹去了。也有一些 PIT 表项一直都等不到匹配的数据包返回,这些表项最终必须因超时而被抹去(这时数据消费者如果仍然坚持想要那份数据,就有责任对那份数据重新表达兴趣)。

当兴趣包到达一个 face 时,一个最长名字匹配操作会被执行。匹配操作所用的索引结构(index)决定了机器会先在 CS 中寻找是否有匹配项,若没有则在 PIT 中寻找,若还没有则在 FIB 中寻找。

如此,若本机的 CS 中已经有匹配兴趣包的数据包,则此数据包会从兴趣包到来的那个 face 被送出,然后兴趣包会被丢弃(因为它已经被满足了)。

如若不然,假如本机有一个 PIT 表项与兴趣包的内容名字完全匹配,则兴趣包到来的那个 face 会被加入到匹配的 PIT 表项中,然后那个兴趣包会被丢弃(因为这种情况表明对于相关数据的兴趣已经从本机向上行表达过了,现在需要确保的就是当对应数据包回到本机时必须向有新兴趣包到来的这个 face 抄送一份)。

若 PIT 中表项无一匹配,但是本机 FIB 中有匹配表项,那么这个兴趣包就必须根据匹配表项被转发。首先,在 FIB 对应表项的 face 列表中去除兴趣包到来的那个 face,若去除之后列表不为空的话,则将兴趣包转发

给列表中的每一个 face。同时,一个新的 PIT 表项要被建立起来,这个表项的索引值是名字,内容是前面所说的 face 列表。

如果在 FIB 中都找不到与兴趣包匹配的表项,那么这个兴趣包会被直接丢弃(因为发生这种情况说明本机既没有现成的匹配数据,也不知道应该向哪里转发去寻找这个数据)。

相对于兴趣包而言,数据包的处理就要简单许多了。数据包不会被路由,它们只是简单地遵从沿路 PIT 表项的指示回到数据申请者(们)身边。当一个数据包到达时,也要做一次最长名字匹配。如果 CS 中有名字匹配则说明在此之前已经有数据包来过,所以这一个数据包就可以被丢弃了。如果有 PIT 匹配(可能不止一项),意味着此节点曾接收过对此数据包的请求。这个数据包可能先被验明正身(详见 3.5.1)然后被加入 CS (即 index 列表中指向该数据包的指针被标记为 C-type)。随后,将所有匹配的 PIT 表项中的请求 face 做集合并,再减去数据包的到达 face,对所得集合中的每一个 face,发出这个数据包。如果存在 FIB 匹配,则意味着 PIT 中没有匹配,或者说此节点从未收到过关于此数据包的兴趣但是数据包确来到了此节点。这个数据包是一个“不请自来”的数据包,应当被丢弃。[5]

[5]“不请自来”的数据包可能由恶意行为产生,也可能是有多个数据源,或者单一数据源但有多条数据传播

节点的内存要用于多路复用，CCN CS 模型允许将内存同时当做网络缓存使用，这点与 IP 的 FIFP 缓存模型不同。这样一来，所有的节点都可以提供缓存，缓存的大小仅仅受节点本身资源丰匮程度和管理政策的限制。

用兴趣包换取数据包这种方法使 CCN 本能地适用于多点通信，这使它在高度动态变化的环境中仍然能轻松维持通信。任何一个对多个网络持有访问权的节点都可以成为网络间的内容路由。一个移动节点通过使用自身缓存可以连通两个原本不相连的网络区域，或通过间歇连接提供延时连通性。 CCN 以此提供对容断网络（Disruption Tolerant Networking）[11] 的支持。兴趣包和数据包的交换在有本地连通性的地方即可进行。例如，两个同事可以用他们的笔记本电脑和自组无线网络在一个与外界网络隔绝的地方正常地分享文件等等。

3.3 传输

CCN 传输被设计为在不可靠包递送服务之上运行，包括在连通性高度变化的移动和普适计算设备中运行。因此兴趣包和数据包都可能在

路径。如果是后两种情况，第一个到达节点的数据包会把兴趣包消费掉，因此后来的数据包将会找不到相应的 PIT 表项。不论是何种情况，为了保证流平衡以及帮助维持网络的平稳运行，“不请自来”的数据包都应当被丢弃。

传输中被丢失或损坏,被请求的数据也有可能暂时无法访问。为了提供可靠的传输服务,在合理时间内未被满足的 CCN 兴趣包必须重传。与 TCP 不同, CCN 的发送者是无状态(stateless)的。在已表达的兴趣无响应的情况下,如果对相关数据仍有兴趣,那么数据的最终消费者(即对数据感兴趣的那个应用程序)应当负责重新表达兴趣。数据接收方的 strategy layer (详见图 3.1)负责重传,同时负责决定用于表达兴趣的通信接口数、同一时间未被满足兴趣的最大容许量、不同兴趣之间的相对优先级等等。

底层的包交换网络可能会使包重复; CCN 的多点散布模式(multipoint distribution)也可能造成包重复。根据 3.2 所述的节点运营机制,所有重复的包都会被丢弃。至此,虽然数据包的散步不会成环,但是兴趣包的散布却可能成环。这样,就有可能存在有的 face 收到了兴趣包,而实际上这个兴趣已经不存在了(被满足了)这样的局面。为了检测和预防此类事件,兴趣包中包含了一个随机数(nonce)。这样,从不同路径收到的重复的兴趣包就可以被丢弃了(见图 3.2)。

CCN 兴趣包使用 TCP ack 包使用的流控制和定序方法。流控制在 3.3.1 中详述。定序在 3.3.2 中陈述。由于节点能见到与它的兴趣包匹配的所有数据这一点是有保障的,对兴趣包的回应时间和回应率可以由

节点直接测量并用于决定对于某些名字前缀的兴趣包应该采取何种策略。这些内容将在 3.3.3 中详陈。

3.3.1 可靠性与流控制

一个兴趣包最多能带回一个数据包。这一基本规则保证了网络中的流平衡,使位于速度迥异的两个网络上的机器也能有效沟通。不过,与 TCP 中一样,可以将数据和请求合并。如果有多个兴趣包要发送,也不必等到第一个数据包回来消费了第一个兴趣包以后才发出第二个兴趣包,而是可以将多个兴趣包一起发出。兴趣包相当于 TCP 的窗口通告(window advertisement)。接收方可以通过调整它发出的兴趣包的数量来动态调整自己的窗口大小。我们在 3.6.2 中展示了这种流水化的效果。由于每一个 CCN 包都是独立命名的,这条流水线不会因为有的包丢失了而堵塞—— TCP SACK 技术所做的事情是 CCN 的本能之一。

在大型网络中, TCP 对话端对端的本性决定了一个事实:即使每一个对话都保持着流平衡,发送方和接收方之间还是有很多地方会因为会话的汇聚(aggregation)而发生拥塞(congestion)。这种拥塞的后果是包的延时和丢失。对此, TCP 的解决办法是让端点动态调整窗口大小,使汇聚流量低于发生拥塞的标准 [20]。之所以需要这样的流控制方法,其根本原因就在于 TCP 的流平衡是端对端的。在 CCN 中,所有通信都是

基于本地的,所以发送方和接收方之间没有一点是不处于平衡状态上的。由于 CCN 的每一跳都在维持流平衡,就没有必要再用其它的技术去控制传输路径中的拥塞了。这与逐跳流控制(hop-by-hop flow control)是不同的。逐跳流控制使用相邻节点间背压(backpressure)来调整连续流的速率; CCN 则没有 FIFO 队列,只有 LRU 缓存。这一设计可以拆解逐跳反馈形成的环,也可以一只振荡。(我们将在未来的一篇论文中详细讨论这个问题)

3.3.2 定序

在一个 TCP 对话中,数据的辨识要依靠序列号。相比序列号而言,CCN 需要一些更复杂的东西: CCN 的消费者们要从海量的数据之中申请一份,并且众多的消费者还可能共享同样的数据。 CCN 对数据的定位和分享依靠的是结构化的、聚合而成的名字。这些名字并不仅仅是一场转瞬即逝的对话中使用到的临时序列号。组成这些名字的至少部分部件是人们可读、可理解的,并且反应了数据的来历信息。虽说 CCN 的名字拥有这些内涵,当它们出现在兴趣包里的时候,它们的通信功能跟 TCP ACK 的功能是一模一样的:道明接受者需要的下一个数据。

在我们细说数据的辨识机制以前,让我们先来谈谈 CCN 的名字。如前文所说,名字是结构化的;每一个名字都由若干个名字部件

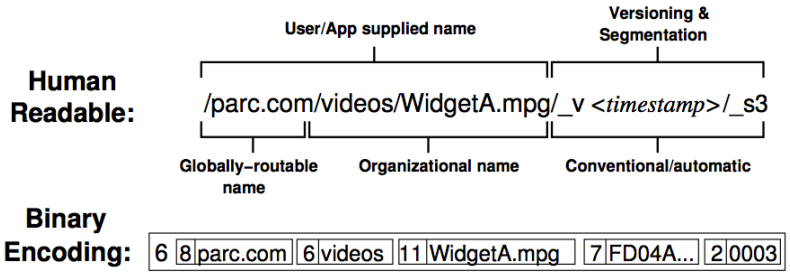

图 3.4 名字示例

(component)组成。每个名字部件又是由若干个八位字节任意数组成的。在此,八位字节任意数是一种对 CCN 通信无影响的变长二进制数。名字必须对协议栈中的高层有意义,但是传输本身不对名字加任何额外的限制,只是要求名字必须结构化,必须由部件组成就够了。整数或其它复杂数值的二进制编码可以直接做名字部件使用,无须专为传输需要转化为文本。名字部件甚至可以出于安全需要而被加密。为了表示方便,我们用类似 URI 的格式书写名字,并用 / 分隔各个名字部件(如图 3.4),但是这些分隔符不是名字的一部分,在编码时也不会被放入包内。图中例子展示了 CCN 应用层面上目前使用的记录内容当下的发展状态的方法(一个版本标记, _v 编码后是 FD,其后跟着一个整数版本号)和分段方法(一个分段编号, _s 编码后是 00,其后跟着的整数值可能是块号,可能是字节号,也可能是该包中视频的第一帧的帧号)。每一个

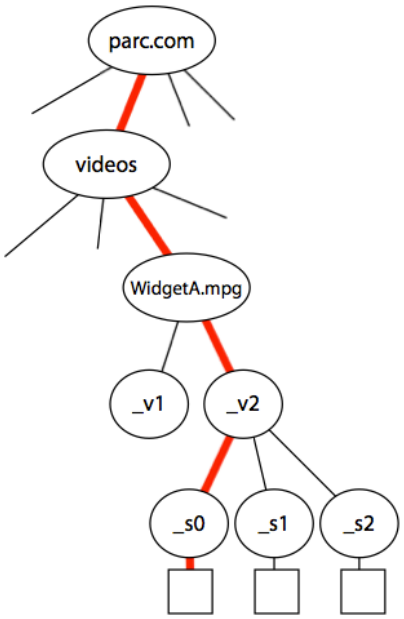

图 3.5 名字树遍历

数据包的名字的最后一个部件隐式地包含了一个对该包的 SHA256 摘要。[6]

兴趣包可以准确地言明它需要什么数据,然而在大多数情况下,下一个需要的数据的全名是未知的,所以消费者根据已知的名字采用“相对命名”法来表达兴趣。这样的做法之所以可行,是由于 CCN 名字树完全可以是有序的(子树按词典编撰顺序排列),从而可以用诸如“下一个”、“上一个”之类的词语来无歧义地表示关系而完全不需要知道名字的语义。

下面举例说明。图 3.5 中展示了名字树中与图 3.4 关联的部分。

[6]摘要部件是可以计算得出的,所以它并不会被传输。文摘的存在可以使兴趣包准确无歧义地指代数据。

如果一个应用程序想要展示最新版视频,它就会表达这样一个兴趣:
/parc.com/videos/WidgetA.mpg *RightmostChild*。在发送方就会执行如图 3.5 中高亮的遍历, [7] 找到该视频的第二个版本的第一段。一旦这份数据被找到了,数据接收方只要发出带有同一名字外加 *LeftmostRightSibling* 注释的兴趣包即可获得后续分段。另一个办法是接收方直接计算出 _s1 部分的名字,因为分段规则是由应用程序制定的,不是由 CCN 制定的。

正如前面例子所展示的,数据命名规则可以根据数据包的相对检索特性而专门设计。应用程序可以通过遍历树发掘存在数据。虽然这些都不属于 CCN 基本传输,但是他们都是 CCN 应用程序设计的重要组成部分。我们期待着一大批可重用的习惯的出现,期待着它们被标准化,期待着它们以共享函数库的方式实现,给应用程序开发者提供高度抽象化的模块(如文件和流媒体的传输)。

兴趣包提供了一种对已有数据的限制性查询机制。它是为高效表达接收方的下一个需求而设计的。在这里我们没有足够的篇幅去赘述我们正在研发的查询选项(query options)的细节。不够我们申明用兴趣包限制数据发布者的输出是可以做到的,不需要将整个数据集或整棵子树

[7]默认的遍历规则是 *LeftmostChild*

都发送出去。在简单排序不够用时,兴趣包还可以排除(exclude)已经收到的包。我们还在研发更高级的名字发掘机制。这些机制可以有效发掘大型子树而不一定要取得内容。

3.3.3 高连通性,移动性及策略

今天的计算机通常都有多个网络接口和极强的移动性。由于 IP 只能在生成树上转发包,它很难利用多网络接口的优势,也很难适应高移动性给网络带来的变化。 CCN 包不能转发成环,所以 CCN 充分利用多网络接口优势。 CCN 谈论数据,而不是对节点谈话,所以它不需要把第三层的身份信息(如 IP 地址)和第二层的身份信息(如 MAC 地址)绑定在一起。即使是在连通性快速变化的情况下, CCN 总是能够快速传输数据——物理传输有多快, CCN 就能做到多快。再者,由于 CCN 的兴趣包和数据包是配对的,每个节点都可以获得每个名字前缀、每个 face 的细粒度性能信息。这些信息可用于动态将名字和 face 匹配(详见 3.6.3)。[8]

正如 3.2 中说的一样, CCN 特意用与 FIB 表项对应的 face 列表模拟了多连通性。由于缺乏能使用所有 face 的万金油式策略,设计的

[8]在 IP 中,路由的不对称性通常决定了中间接地啊不可能知道某一个接口是否真的工作良好,因为路由只能看到一个对话的一个方向。

初衷是让每个 FIB 表项都包含一个程序。这个程序像一台抽象机器，专门负责决定怎样转发兴趣。这台机器的“指令”应该包含普通存取、算术、比较指令的子集，包含一些动作（*sendToAll, sendToBest, markAsBest*）和触发器（*interestSatisfied, interestTimedOut, faceDown*）。与此匹配的，face 会有一套开放式的属性（*BroadcastCapable, isContentRouter, UsageBasedCharging, PeakUseLimited*）。使用动作，属性可以被动态创建。

这些动作、触发器和属性合并在一起统称策略层（CCN Strategy Layer）。FIB 表项中的程序就是获得这个表项所指数据的策略。我们目前的默认策略是先给所有具广播能力（*BroadcastCapable*）的 face 发兴趣包，如果没有回复，则依次尝试所有其它 face。本地数据（如一场报告中报告人机器中的数据，一次商业会议中同事电脑或手机中的数据）可以直接获得。只有在本地找不到的数据才会使用路由机器去获取。

FIB 表项中的 face 信息可以通过各种方式获得。如图 3.6 中的数据源可以用 *Register* 动作作用于本地 CCN 内核，接收对它可以提供的数据的兴趣。这么做会在本地 FIB 表中创建起相关兴趣的表项，并且会在它们后面填上对应程序的 face。已注册的名字会有一个标志位，决定它们是否应该被宣传到本地机器之外。宣传代理（announcement

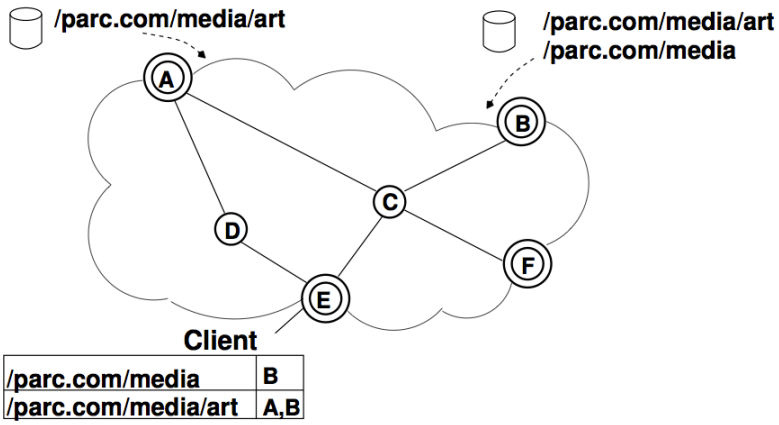

图 3.6 将兴趣包路由至域内多媒体内容存放处

agent)扫描本地注册名字表,把负责他们要求的、被置位了的名字广而告之(详见 3.5.4)。本地扫描可以使用局限于本地名字空间的正常 CCN 兴趣包和数据包来做。广告可以使用 CCN (即让宣传代理提供 `/local/CCN/registrations`),可以使用标准 IP 服务位置协议(SLP),或使用 CCN 或 IP 路由(详见 3.4)。

3.4 路由

近年来路由在学术研究领域的热度回升。今天大多数路由问题都有了许多有趣且有效的解决方案。所有对 IP 奏效的路由机制都应该对 CCN 有效,因为 CCN 的转发模型是 IP 模型的严格超集。 CCN 转发模型有更少的限制(不为了避免成环而禁止多数据源、多目的地传输),并且路由语义(结构化名字集合与最长匹配)也与 IP 的转发模型相同。

CCN 为路由协议的传输提供了一个极佳的交通工具:大多数路由传输协议的核心是一个与 CCN 的内容为心、有引导的弥漫扩散模型极为相似的东西。此中原因是它们的工作时间都在网络的拓扑期之前,这时候对等体双方的位置和身份都还未知。由于 CCN 是一个文件的信息安全模型(详见 3.5),使用 CCN 进行路由传输即可自动完成路由保护。

为了解释如何将 CCN 映射到路由机制中去, 3.4.1 描述了 CCN 将如何使用未经任何修改的互联网链路状态内部网关协议(IS-IS 或 OSPF)路由。这既是为了展示 CCN 如何使用现有常规路由方法, [9] 又是为了说明 CCN 与 IP 的兼容性极好,使用现有的基础设施即可大范围使用。

3.4.1 链路状态域内路由

域内路由协议为节点提供了发现和描述本地连通性(邻接性)和描述直接连接着的资源(即宣告的内容)的途径 [17, 16]。这两种方法是正交的——一个描述图中的链接,另一个描述图中节点的拥有的资源。这两个操作常常在完全不同的信息域中完成。比如, IS-IS [17] 用 IEEE 802.1 第二层的物理地址描述连通性,而使用第三层 IP4 或 IP6 前缀来宣告内容。如 3.2 中所说的, IP 转发和 CCN 转发基本上是相同的。它们

[9]现有传统路由方法是相对于那些以内容为导向的路由方法(如 Small Words)而言的。那些方法也可应用于 CCN,但是采取的策略完全不同。

都使用基于前缀的最长匹配(而且使用的理由也一样——结构化的细节集成)来找拥有更接近于辨识标志的邻居。既然两个 FIB 有诸多相似之处,读者可能会猜想那些用于创建 IP FIB 的设备也许可以轻松地被改造成为创建 CCN FIB 的设备。事实正是如此。

CCN 前缀与 IP 前缀是不同的,所以主要的问题是它们是否可能被一些路由协议描述和表达。幸运的是, IS-IS 和 OSPF 都可以用一个通用的类型标签值(TLV, type label value)机制 [18, 19] 来描述直接连接的资源,并且这个机制对于散布 CCN 内容前缀也很合适。此机制的明细中说,无法识别的类型应该被忽略。这就是说实现 CCN 全套转发模型的路由可以直接依附到现存的 IS-IS 或 OSPF 网络中去,无需改动现存网络或者它的路由。内容路由通过邻接协议学习到网络物理拓扑结构,并在那个结构中宣告自己的位置。路由还可以通过 CCN TLV 机制来宣告内容。

举个例子,图 3.6 展示了一个含有一些(单周期) IP 专用路由和 IP CCN 复合路由的内部网关协议(IGP)域。靠近 A 的一个多媒体仓库正在(通过向本地网络管理名字空间发送广播的形式)宣告它可以为与前缀 `/parc.com/media/art` 匹配的兴趣包提供服务。 A 中的路由程序听见了这个宣告(因为这个路由程序已经对其名字空间内的这类宣告表达过兴趣),于是它就在本地 CCN FIB 为此前缀新建

一个表项,在 face list 中填上它收到宣告的那个 face。这个路由程序随后把这个前缀封装为 IGP LSA。这样这个前缀就扩散到网络所有节点。当网络中的其他路由程序,比如说 E 中的路由程序,首次得到这个 LSA 时,它会新建一个到 A 的 CCN face,并在本地 FIB 中添加 /parc.com/media/art 一项, face list 写 A。当一个靠近 B 的仓库宣告 /parc.com/media 和 /parc.com/media/art 时, B 也将这两个前缀的 IGP LSA 扩散出去,其结果是 E 的本地 CCN FIB 会如图 3.6 中所示的那样。如果 E 的附近有客户表达 /parc.com/media/art/impressionist-history.mp4 这样的兴趣,这个兴趣会向 A 和 B 转发,而 A 和 B 则各自发送给自己附近的多媒体仓库。

CCN 动态创建对于带宽和延时而言近似最优的拓扑结构。数据至向有兴趣存在的地方流动,且选择最短的路径。对于任何一份数据,任何一个连接中至多只有一份它的副本经过。然而这种递送拓扑目前显然不是最优的,因为如果靠近 F 的客户也对同一部电影感兴趣的话那么同样内容的第二个副本又必须流过 A-C 连接或 B-C 连接。这种情况发生的原因是在 CCN 部署过程中有很多拓扑结构对于 CCN 而言不可用(C 不是内容路由,所以它不会缓存数据)。一旦 C 进行了 CCN 软件升级, E 和 F 就会通过它来转发兴趣,那么传输就是最优的了。

在上面讨论过的模型中, IGP LSA 被使用于传输含有完整的 CCN 验证、保护和政策注释的标准 CCN 信息。因此即使 IGP 是不安全的,存在于具有 CCN 能力的节点间的通信也是安全的。如果所有节点都具有了 CCN 能力,那么 IGP 拓扑结构自然就安全了(详见 3.5.1)。保障外界产生的前缀布告的安全性也是宣告协议的功能之一。 CCN 内容前缀,如图 3.6 中媒体服务器宣告的那些,有稳健的信任模型,并由 CCN 保障其安全性。外界 IGP 或 BGP(边界网关协议) 产生的 IP 前缀将不被信任。

当存在多个同一前缀的布告时, IP 和 CCN 之间存在行为差异。在 IP 网中任何所有节点都会把所有匹配的通信量送到一个宣告者那里去。在 CCN 网中所有节点会把所有匹配的兴趣包发到所有宣告者那里去。这一行为差异的根源在于语义差异:从某个 IGP 路由来的 IP 前缀布告会说“所有带有这个前缀的主机都可以通过我访问到”;与之等价的 CCN 布告会说“一些带有这个前缀的内容可以从我这里访问到”。由于 IP 没有办法在内容层面上防止成环,它只好去建设无环转发拓扑结构,即一个以目的地为根节点的汇点树。由于一棵树中的任意两节点间只有一条路径,一个 IP FIB 表项只有一个用于存放输出接口的空位。于是所有与某个前缀关联的主机都必须通过发出布告的这个节点

才能访问,因为所有的通信量都被送到那里去了。由于 CCN 包无法成环,一个前缀布告不需要保证某一节点与前缀所能表示的所有内容都相连,且 CCN FIB 也被设计成将兴趣转发至宣告过前缀的所有节点。这一语义变迁可以在不改变 IGP 的前提下完成,因为这是实现上的变化,不是协议上的变化。 IP 要计算从前缀宣告计算生成树, CCN 却不要。CCN 是在信息消费处计算,而不是在信息生产处计算,所以它和 IP 最终收到的信息都是完整的。[10]

3.4.2 域间路由

一旦一个互联网供应商(ISP)的一部分客户开始使用 CCN,ISP 就最好开始部署内容路由器以节省成本(一份数据不管有多少用户需要,都只需要从互联网供应商的连接中走过一次)和降低客户平均延时(一份数据的多个副本除了第一份之外均来自互联网供应商的本地内容存储库)。因此,一旦 CCN 的部署冲破一个基线以后,它就会自底向上地,以一种激励性的方式迅速成长。由于用户直接与他们的供应商相连,他们没有必要借助一个服务发现协议来了解供应商的内容路由。这类协议可以把 CCN 注册公告(详见 3.3.3)当做通配符(零前缀),或者使用任拨

[10]严格地说,这句话对链路状态内部网关协议如 IS-IS 和 OSPF 是正确的,但对距离矢量协议如 RIP 和 EIGRP 却不正确。后者的路由公告涉及到 Bellman-Ford 计算,预测生成树并抑制其它连接。这样的 IGP 就需要稍稍改动一下文中所述的机制了。

(比如所有内容路由的 IP 地址都设为 10.0.96.95), DNS 惯例(如所有内容路由都使用 *ccn.isp.net* 域名), DNS SRV [14], SLP [15] 等等。这些选项都不要求散布任何内容前缀。

这样自底向上的部署方法最大的问题在于跨越被无内容路由 ISP 分割的有内容路由域之间的鸿沟。比方说一个位于 *parc.com* 的内容路由想要前缀为 *mit.edu* 的内容,然而 PARC 与 MIT 之间并无内容路由器。使用此前缀进行一个(启发式的) DNS 查找,确定位于 MIT 的内容服务器(群)的 IP 地址(或通过 *_ccn._udp.mit.edu* SRV 查找,或进行 *ccn.mit.edu* 地址查找),可以自动地、按需地建立起连接两个内容服务器的基于 UDP 隧道的 face。然而当鸿沟并不位于两个域的边缘时,这一机制就工作得不那么理想了。如果 PARC 和 ISP 的供应商都提供内容路由服务但是它们却由不支持 CCN 的供应商连接, PARC 的供应商就无法获知 MIT 的供应商那里的相关路由,于是它就会把兴趣包发送到 MIT。因此若不添加额外机制, ISP 路由器照顾了进口内容(被它的客户要求过的内容),却没有照顾到出口内容(它的客户生产的内容)。这在一定程度上削弱了 CCN 的一个主要的长期优势,即使内容散布树根节点处的通信量与内容流行程度无关。今天,这个通信量仍在随着流行程度增长(详见 3.6.2)。

这一问题可以通过把域级内容前缀整合入边界网关协议(BGP)来解决。目前的 BGP 域间路由已经有 IGP TLV 的等效机制,允许域广告其客户的内容前缀。 BGP 的 AS-path 信息亦可以使各个域建立起在自制系统层面上的而非网络前缀层面上的拓扑地图。这个地图在功能上与 IGP 建立的地图等效(它使人获知提供某个前缀的域,以及通向这些域的最短路径上有哪些具有 CCN 能力的域),所以算法可以通用。

3.5 内容为心安全

CCN 建立在基于内容的安全之上:保护和信任与内容一路随行,不是内容经过的连接的一种属性。在 CCN 中,所有的内容都由数字签名验证;私密内容由加密算法保护。这是 CCN 动态内容缓存机制的一层重要保障——如果你将要得到一份离你最近的数据副本,你必须能够验证你得到的数据。目前的 IP 网络根据内容从哪里来和内容是怎样得到的来决定是否信任内容,因此客户必须从原始数据源那里直接获取内容。使安全与内容融为一体可以减少我们在网络中间环节的忧虑,从而开放网络获得广泛参与。在这个章节中,我们给出 CCN 核心安全设计的总览,并强调其中的新颖之处。详细的分析,包括撤销之类的话题,都要在另一篇文章中讨论。额外背景和研究动机在 [34] 中描述。

3.5.1 内容验证

CCN 验证名字与内容之间的绑定。每个 CCN 数据包的签名(如图 3.2)都是对该数据包的名字、内容和少量对签名验证有帮助的辅助信息(图 3.2 中的 signed info)进行的。这使内容发布者可以安全地将任意名字与内容绑定。与此形成对比的是,此前的许多方法为了安全,都要求名字能够自认证(self-certifying),如使用内容的加密摘要作为它的名字 [16, 28, 12, 9]。直接使用对用户或对应用程序有意义的名字的能力增强了可用性,简化了传输。没有这种能力的系统将需要一个“间接设备(indirection infrastructure)”[4, 6] 把人们关心的名字映射到安全的、不透明的、自认证的名字上去。这样的系统的安全性就受那个(常常是不安全的)间接设备的限制。

CCN 数据是可以公开验证的——每个数据包的签名都是标准的公钥签名,并且通信流上的任何机器(包括端点和沿路的所有机器)都可以验证一个名字和内容的绑定是否是用某个公钥签署的。签名算法是有内容发布者从一个大的确定的集合中根据某一数据的性能要求选择的——比如要求验证数据最少,或要求签名的生成和验证的延时和计算成

本最小等。虽然数据包都被设计为单独验证,我们仍然可以用像 Merkle Hash Trees [27] 这样的集成办法降低计算成本。

每个签了名的 CCN 数据包都含足够的信息使人能够以此获取到验证此包的公钥。它的支持信息中包含有那个公钥的加密文摘。它可以作为数据发布者的一个简短辨识符号,也可以帮助验证者快速地从本地缓存中找到那个公钥。包中还含有一个钥匙定位符,指示了获得公钥的方法;它可能是公钥本身,也可能是能够用于获取公钥的 CCN 名字。

CCN 最底层的内容验证完全是语法上的验证——它只是确认一下内容是由它声称的那个公钥签署的。它并不给那个公钥赋予任何实际意义——它属于谁,它所签署的数据是否应该被用户信任。即便是这样一点微不足道的验证工作也是很有用的,尤其是在防御多种网络攻击的时候。例如,这使得内容消费者在申请数据时既指定内容的名字又指定内容的发布者,这样他们就可以在有虚假恶意信息存在的情况下仍然得到他们想要的信息。 CCN 的路由器可以选择全验证、部分验证或不验证它们经手的数据包,这视它们的资源数量而定。它们还可以根据是否检测到网络攻击动态地决定是否验证更多的数据。

3.5.2 信任管理

虽然 CCN 以点对点形式传输数据,它却在数据发布者和数据消费者之间提供端对端的安全。 CCN 内容消费者必须决定收到的内容是否是值得信任的。在 CCN 中信任的概念是上下文相关的,即它是由内容的上下文和内容的用途决定的。比如,我们可能需要一份法律文件由被法庭授权的人签署,一篇博文则只需要签上与这个博客里的其他文章一样的签名就可以了,有时可能连这个要求都不需要达到。这种方法较万金油式的信任管理办法(比如认定数据发布者非黑即白)更为灵活易用。

基于内容的安全的基本原语——经验证的从名字到内容的绑定——可以用来实现更高级的信任机制。 CCN 签名绑定机制的本质作用是认证内容。当名字指代一个组织或个人,且内容是一个公钥时,这就是一个数字证书。这使 CCN 能够很容易地支持传统的密钥信任建立机制。更有趣的是,通过允许内容安全链接到另一个内容,我们就可以用内容去认证内容。这样我们就可以把对少量密钥的信任转化为对一群互相关联的内容的信任。

3.5.2.1 公钥信任

应用层面的 CCN 消费者必须面对传统的密钥管理问题——将公钥与个人和组织相关联(因为通常情况下都是这些现实生活中的身份信息在决定谁是一个内容的可信签署者)。 CCN 从几个方面将此任务简化了:第一,它直接、简单地解决了获取公钥这个实际问题。密钥只是另一种形式的 CCN 数据; CCN 简单的命名习惯确保了它们能够容易地被找到。

第二,如上文所说,只需要把公钥作为 CCN 内容发布就相当于为公钥生成了一张证书—— CCN 名字和那个公钥的绑定是由发布者(签名者)认证的。这个积木式部件可以来表示任意密钥之间的信任关系——从简单的树(如传统的公钥基础设施, PKI)到任意的图(如加密隐私信任网 PGP Web of Trust)。

第三, CCN 不强制使用任何普适的信任模型。信任是内容发布者和消费者之间的事。对一个应用程序合适的可能对其它程序并不合适。用户可以自由使用现存的公钥信任模型(如 PKI),或者自定义新的适合 CCN 使用的模型。

SDSI/SPKI [32, 10, 2] 的模型就特别适合 CCN 使用。在这个模型中公钥和现实身份由本地控制的命名空间映射。比如一个组织的成员的

公钥如果被这个组织所证明,那么这个成员就可以被承认,无需通过外界第三方来证明(如 Verisign)。知道了 parc.com 的公钥,我们就可以验证公司员工的公钥。更进一步,如果我们信任 parc.com 的一个员工,那么我们可以去他的 SDSI 名字空间查看 parc.com 的身份和公钥。使用一些用户友好的机制(如个人通讯录,组织成员名单,公众经验等 [30, 37]),我们可以得到一个公钥已经验证的用户集合。从这个很小的集合出发,我们可以用 SDSI 模型推演出一个我们信任的内容发布者的大集合。

我们可以将 SDSI 身份直接映射为 CCN 名字,并且直接在 CCN 内容里表达 SDSI 信任关系。这样的名字空间树可以组成一片森林——一个内容消费者可能会相信她拥有 parc.com(或者 /parc.com/george)的真实公钥;她这么相信的原因可能是由于她的亲身经历(比如她是 PARC 的一个员工),可能是基于她的朋友提供的信息,也可能是由于此公钥出现在受信任的公钥目录中。 CCN 不要求,甚至不预想这些树有一天会像它们在传统的全球商业 PKI 中一样聚集成一棵(或只有少数几个根节点)。最值得注意的是,是消费者们根据各种信息决定他们为什么要信任一个公钥,而不是依靠发布者们从某一个商家那里获取一个证书。

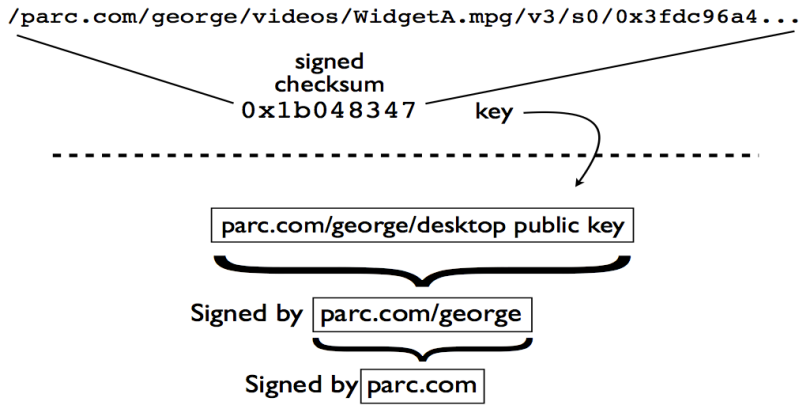

图 3.7 CCN 信任建立模型可以将内容空间与发布者密钥相关联

另外,通过把内容组织在层次化的名字空间里, CCN 可以把签名政策甚至是公钥与特定的名字关联起来;一个层级的名字空间可以由它的上一级来签名认证。如图 3.7 所示, `parc.com` 的密钥认证了用户 `george` 的密钥;这个密钥又认证了 `george` 那台桌面电脑的密钥。这些以 CCN 数据形式存在的信任宣言可以帮助消费者评估此发布者究竟是不是 `WidgetA.mpg` 的可靠发布者。

3.5.2.2 基于证据的安全

我们可以在结构化名字的概念中添加安全引用(secure reference)一词。它与受信任的超链接或书签很相似。一个 CCN 内容不仅可以使用名字来引用另一个内容(引用目标),它还可以使用加密文摘(加密文摘在功能上也是一种能够自认证的名字 [26, 28, 12, 9])或使用内容发布者的身份信息(密钥)来引用 [9, 31, 25, 24]。这样的引用可以用来表达委托

(delegation)的概念。比如一个数据发布者 P 发布了一个名为 N 的链接,引用目标为 (N', P'),这意味着链接发布者 P 希望 N 指代 P' 发布的 N' 的引用目标。

这样的引用引用引用可以表达传统形式的委托,也可以用于建立一个内容信任网络——经签名认证的内容个体有效地认证它们安全连接到的内容,组成一张网络。比如如果用户决定了信任一个名为 A 的网页,那么他就可以自动地信任由 A 安全连接到的内容,如它的图片、广告、原材料等等,不需要额外的管理和设置。这样的信任的粒度是极细的——那些材料只是在 A 的上下文关系下才被认可。

用户遇到的每一个内容都是它的引用目标的可靠性的一个潜在证据。如果很多我们信任的发布者都说它们信任 P' 给出的 N' 的值,那么我们就更有可能相信这个值。如果一个黑客袭击了一个发布者,比如获取到了 P'' 的私钥并用它伪造了一个 N' 的值,这样的攻击将不会奏效,因为大多数证据仍然指示出了正确的值。每增加一份可信的签名证据,这样的黑客袭击就会难上加难,因为袭击者是无法销毁所有证据的。

3.5.3 内容保护与访问控制

控制 CCN 内容访问的首要方法是加密。 CCN 不强制受信任的服务器和目录使用访问控制协议;不管是谁得到了私密的内容,只有经授权的用户才能解密。

对数据、名字、名字部件的加密都是对网络完全透明的——对 CCN 而言,它们都不过是二进制数据而已(尽管出于路由的高效性和数据排序的角度考虑一些名字部件最好是明码)。解密密钥可以以 CCN 数据块的形式和它们的内容一起散布。封装在用户友好的函数库中的命名习惯可以使查找解密密钥的任务变得容易。 CCN 不强制使用任何加密机制或密钥配送机制——用户只需要为一个内容确定编码和散布解密密钥的方式就可以建造起自定义的、面向应用程序的访问控制模型。

3.5.4 网络安全与措施强化

3.6 评估

3.6.1 数据传输的效率

3.6.2 内容散布的效率

3.6.3 VoCCN 与策略层

3.7 相关工作

3.8 结论

鸣谢

我们感谢 Paul Stewart, Paul Rasmussen 和 Simon Barber 对论文实现的大力帮助。我们还要感谢 Ignacio Solis, Marc Mosko, Eric Osterweil 和 Dirk Balfanz 的富有成效的讨论的建议。

3.9 参考文献

[1] Project CCNx™. `http://www.ccnx.org`, Sep. 2009.

[2] M. Abadi. On SDSI's Linked Local Name Spaces. *Journal of Computer Security*, 6(1-2):3–21, October 1998.

[3] B. Adamson, C. Bormann, M. Handley, and J. Macker. *Multicast Negative-Acknowledgement (NACK) Building Blocks*. IETF, November 2008. RFC 5401.

[4] W. Adjie-Winoto, E. Schwartz, H. Balakrishnan, and J. Lilley. The Design and Implementation of an Intentional Naming System. *SIGOPS Oper. Syst. Rev.*, 33(5):186–201, 1999.

[5] A. Anand, A. Gupta, A. Akella, S. Seshan, and S. Shenker. Packet Caches on Routers: The Implications of Universal Redundant Traffic Elimination. In *SIGCOMM*, 2008.

[6] H. Balakrishnan, K. Lakshminarayanan, S. Ratnasamy, S. Shenker, I. Stoica, and M. Walfish. A Layered Naming Architecture for the Internet. In *SIGCOMM*, 2004.

[7] M. Caesar, T. Condie, J. Kannan, K. Lakshminarayanan, I. Stoica, and S. Shenker. ROFL: Routing on Flat Labels. In *SIGCOMM*, 2006.

[8] D. Cheriton and M. Gritter. TRIAD: A New Next-Generation Internet Architecture, Jan 2000.

[9] I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong. Freenet: A Distributed Anonymous Information Storage and Retrieval System. *Lecture Notes in Computer Science*, 2009:46, 2001.

[10] C. M. Ellison, B. Frantz, B. Lampson, R. Rivest, B. M. Thomas, and T. Ylonen. *SPKI Certificate Theory*, September 1999. RFC2693.

[11] S. Farrell and V. Cahill. *Delay- and Disruption-Tolerant Networking*. Artech House Publishers, 2006.

[12] K. Fu, M. F. Kaashoek, and D. Mazières. Fast and secure distributed read-only file system. *ACM Trans. Comput. Syst.*, 20(1):1–24, 2002.

[13] J. F. Gantz et al. IDC - The Expanding Digital Universe: A Forecast of Worldwide Inform ation Growth Through 2010. Technical report, March 2007.

[14] A. Gulbrandsen, P. Vixie, and L. Esibov. *A DNS RR for specifying the location of services (DNS SRV)*. IETF - Network Working Group, The Internet Society, February 2000. RFC 2782.

[15] E. Guttman, C.Perkins, J. Veizades, and M. Day. *Service Location Protocol*. IETF - Network Working Group, The Internet Society, June 1999. RFC 2608.

[16] IETF. RFC 2328 – OSPF Version 2.

[17] IETF. RFC 3787 – Recommendations for Interoperable IP Networks using Intermediate System to Intermediate System (IS-IS).

[18] IETF. RFC 4971 – Intermediate System to Intermediate System (IS-IS) Extensions for Advertising Router Information.

[19] IETF. RFC 5250 – The OSPF Opaque LSA Option.

[20] V. Jacobson. Congestion Avoidance and Control. In *SIGCOMM*, 1988.

[21] V. Jacobson, R. Braden, and D. Borman. *TCP Extensions for High Performance*. IETF - Network Working Group, The Internet Society, May 1992. RFC 1323.

[22] V. Jacobson, D. K. Smetters, N. Briggs, M. Plass, P. Stewart, J. D. Thornton, and R. Braynard. VoCCN: Voice-over Content-Centric Networks. In *ReArch*, 2009.

[23] C. Kim, M. Caeser, and J. Rexford. Floodless in SEATTLE: A Scalable Ethernet Architecture for Large Enterprises. In *SIGCOMM*, 2008.

[24] T. Koponen, M. Chawla, B.-G. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, and I. Stoica. A Data-Oriented (and Beyond) Network Architecture. In *SIGCOMM*, 2007.

[25] J. Kubiatowicz et al. OceanStore: An architecture for global-scale persistent storage. *SIGPLAN Not.*, 35(11):190–201, 2000.

[26] D. Mazières, M. Kaminsky, M. F. Kaashoek, and E. Witchel. Separating Key Management from File System Security. In *SOSP*, 1999.

[27] R. C. Merkle. *Secrecy, authentication, and public key systems*. PhD thesis, 1979.

[28] R. Moskowitz and P. Nikander. *Host Identity Protocol Architecture*. IETF - Network Working Group, May 2006. RFC 4423.

[29] B. Ohlman et al. First NetInf architecture description, April 2009. `http://www.4ward-project.eu/index.php?s=file_download&id=39`.

[30] E. Osterweil, D. Massey, B. Tsendjav, B. Zhang, and L. Zhang. Security Through Publicity. In *HOTSEC*, 2006.

[31] B. C. Popescu, M. van Steen, B. Crispo, A. S. Tanenbaum, J. Sacha, and I. Kuz. Securely replicated web documents. In *IPDPS*, 2005.

[32] R. L. Rivest and B. Lampson. SDSI - A Simple Distributed Security Infrastructure. Technical report, MIT, 1996.

[33] M. Särelä, T. Rinta-aho, and S. Tarkoma. RTFM: Publish/Subscribe Internetworking Architecture. In *ICT-MobileSummit*, 2008.

[34] D. K. Smetters and V. Jacobson. Securing network content, October 2009. PARC Technical Report.

[35] I. Stoica, D. Adkins, S. Zhuang, S. Shenker, and S. Surana. Internet Indirection Infrastructure. In *SIGCOMM*, 2002.

[36] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan. Chord: A Scalable Peer-To-Peer Lookup Service for Internet Applications. In *SIGCOMM*, 2001.

[37] D. Wendlandt, D. Andersen, and A. Perrig. Perspectives: Improving SSH-style host authentication with multi-path probing. In *USENIX*, 2008.