Catching popular prefixes at AS border routers with a prediction based method

Wei Zhang\textsuperscript{a,c,*}, Jun Bi\textsuperscript{a,b}, Jianping Wu\textsuperscript{a,b}, Baobao Zhang\textsuperscript{a}

\textsuperscript{a} Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China
\textsuperscript{b} The Network Research Center, Tsinghua University, Beijing 100084, China
\textsuperscript{c} National Academy of Defense Information, Wuhan 430010, China

\textbf{A R T I C L E   I N F O}

\textbf{Article history:}
Received 19 January 2011
Received in revised form 21 December 2011
Accepted 6 January 2012
Available online 12 January 2012

\textbf{Keywords:}
BGP
Popular prefix
Route caching
Virtual aggregation

\textbf{A B S T R A C T}

Modern Internet routers require powerful forwarding facilities to cope with extremely high rate Forwarding Information Base (FIB) lookups. In general, the FIB is constrained to a small highly efficient but expensive memory. Unfortunately, the BGP route table (RIB) keeps increasing, and this subsequently results in severe FIB inflation at BGP routers. What if we only load a small portion of the RIB into the FIB? Recently the route caching mechanism has been revisited. With such a route caching mechanism, the optimal method is to load in a FIB with popular prefixes which contribute major traffic loads. We propose a prediction based method to catch those popular prefixes with a limited cache size.

In this paper, the dynamics of popular prefixes has been studied based on real traffic traces from different ISPs. On applying a GM(1,1) model which is widely applied in grey system control and prediction, we propose a traffic prediction-based route caching method which attempts to bias the cache dump strategy with a range of history to ameliorate the effects of bursts from non-popular prefixes. We also suggest applying FIB aggregation techniques, e.g. Optimal Routing Table Constructor (ORTC) algorithm, to suppress the number of non-popular sub-prefixes of the popular prefixes on route updates. The evaluation of our method is based on simulation over real traffic traces. The simulation shows our prediction-based cache replacement strategy outperforms other cache strategies and matches Internet traffic dynamics very well.

© 2012 Elsevier B.V. All rights reserved.

1. Introduction

Internet core routers are facing challenges brought by the ever-increasing transit bandwidth and routing scale. The Autonomous System border routers (ASBRs) may have to process packets on each port at a rate of 40 Gbps. This requirement can only be met by applying high-performance hardware, such as TCAM or SRAM. However, this kind of special-purpose memory is costly and energy eager while normally its size can not be very large. On the other hand, the size of BGP RIB increases inexorably. If we load every entry of the RIB into the FIB, a router is under great pressure of FIB inflation and may sometimes be at a risk of memory overflow on its forwarding panel.

Recently there have been some new proposals to address this routing scalability problem. Although there might be different solutions to tackle this problem, the most straightforward method that can be thought of is to reduce the FIB size directly. The ViAggre virtual aggregation technique defines a distributing forwarding scheme within an Autonomous System (AS)\textsuperscript{[1]}. Each router only needs to load a small portion of RIB into the local FIB, and this portion should be technically covered by a Virtual Prefix (VP). These VP-assigned routers are Virtual Prefix Routers (VPRs). Each VPR advertises its assigned VP. Therefore, the whole RIB of an AS is distributed into a VPR
community. Every router has a FIB which holds a portion of the detailed routes (entries in the RIB) in addition to a couple of VPs. When a packet needs to be forwarded, if the router has the exact route in its FIB (conventionally a FIB hit), the packet is forwarded promptly; otherwise (namely a FIB miss), the packet will be tunneled to a corresponding VPR who knows a correct route to the destination. This scheme substantially reduces FIB size if the VPR community has enough members. However, this gain is not free of charge.

In ViAggre, a FIB miss will introduce forwarding path stretch and sometimes incur influx traffic to a busy VPR. In order to improve the routing performance, ViAggre suggests loading some additional popular prefixes into the FIB. The popularity of a prefix is determined by the hitting rate of working traffic loads on it. In the rest of this paper, we continue to use the conception of popular prefixes as defined in ViAggre.

Another scenario is the OpenFlow [18] application, which also concerns effective route cache mechanism. OpenFlow proposes to decouple the functions of forwarding and routing. Switches or other forwarding devices will be controlled by a concentrated controller which sends forwarding instructions in the form of flow tables to OpenFlow enabled forwarding facilities. Since most forwarding devices cannot maintain a large FIB, the entries in a flow table have to be dumped if they are expired. In other words, the switch boxes cache partial routes of the RIB. Any unknown flow (not defined in a flow table) will be handed to the controller. The controller will become a bottleneck if there are too many packets that need to be handled. In this case, the efficiency of the routing system would be improved if the caching strategy guarantees popular prefixes always stay in the flow tables of the forwarding boxes.

In both of the two techniques (VA and OpenFlow), we can abstract the following route caching model. The data plane (FIB) only caches a portion of routes of the control plane (RIB). A cache hit means a forwarding device has the route in its FIB and knows how to forward the incoming packet correctly. On the other hand, a cache miss means a forwarding device does not have an effective forwarding route in its FIB; consequently it will forward the packet to somewhere the packet can be handled or retrieve routing information from the control plane. Thereby, the forwarding will be delayed in case of a cache miss. As optimization of the route caching mechanism, our objective is to minimize the forwarding delay or the rate of cache misses.

Motivated by this consideration, we propose to take advantage of the feature of the real traffic, and pursue to cache the right popular prefixes in the FIB to maximize the hitting rate. However, how to catch the popular prefixes with the working traffic dynamics is challenging.

Among other things, the caching strategy is a key issue. A route caching strategy concerns the size of the cache, the cache miss rate in terms of packets and other corresponding overheads incurred by route updates. This paper studies the dynamics of popular prefixes in BGP with regard to inter-domain traffic and proposes a pragmatic route cache strategy for AS border routers. In order to minimize the miss rate at a given cache size, we propose a prediction-based method, and our route cache replacement strategy will be only guided by the prediction. The prediction algorithm is derived from a GM(1,1) model which is widely used in the grey system control and prediction. In other words, we assume the Internet traffic trace has some grey information which will guide our further cache decision effectively. The evaluation shows that our strategy matches the dynamics of popular prefixes very well and outperforms other traditional replacement strategies such as least recently used/least frequently used (LRU/LFU) strategies. Our method can apply sampled traffic traces. In general, the storage and computational overheads are acceptable.

When loading a popular prefix into the FIB, there may be numerous more specific (covered) sub-prefixes that also need to be loaded otherwise the forwarding might be incorrect. This problem has been one of the main challenges in this area. The challenge here is not the load, but the number of non-popular sub-prefixes that must come along with the covering prefix. For example, including the AT&T/8 prefix in the cache would require adding hundreds of additional prefixes into the cache. To get around this problem, we propose optionally to apply FIB aggregation techniques to reduce the number of sub-prefixes covered by a popular prefix. An Optimal Routing Table Constructor (ORTC) [20] algorithm has been implemented and most overlapped prefixes can be suppressed. Our simulation shows that the overheads are generally acceptable even in the worst case.

The evaluation is based on simulation with different ISPs’ traffic traces. In addition to the working traffic traces collected by China Telecom and CAIDA, we also use traffic trace collected from CERNET as a representative of small ISPs. The outcome of the simulation shows that although the popularity of a prefix can be different from ISP to ISP, most of the popular prefixes are predictable if guided by our prediction-based method; non-popular prefixes can be bursty sometimes, but only contribute a small fraction of the overall traffic load; our prediction-based cache method leverages the route caching mechanism efficiently and outperforms static or other cache-miss-based route caching strategies. The GM prediction can retrieve effective information from the sampled data, and more importantly, not much sensitive to the sampling ratio. The overheads of our method are generally acceptable if the sampling ratio and cache update interval has been set properly. The FIB compressing technique can be optional when the gain is obvious. Furthermore, an additional computing device (e.g. PC) can be deployed to help the router on processing the statistics and prediction jobs, since those tasks are independent of primary routing functioning.

The rest of the paper proceeds as follows. Section 2 reviews related works. Section 3 discusses route caching and relevant cache strategies. Section 4 analyzes the feature of Internet traffic and introduces the grey prediction model we use in our traffic prediction mechanism. Section 5 illustrates our prediction based cache replacement strategy in detail as well as the ORTC implementation. Section 6 specifies our methodology of simulation and evaluates the performance and overheads of our method. In the end, Section 7 draws a conclusion of our work.
2. Related works

In the scheme of Virtual aggregation (VA), ViAggre [1] proposes to load popular prefixes to improve VPRs’ performance. ViAggre uses a static set of popular prefixes measured on the granularity of ISP/Point of Presence (PoP). Their experiments show that loading a relatively static popular prefix set can improve the performance undoubtedly. However, they assume the set of popular prefixes is remarkably stable and can be set statically. The authors of VA admit their research is rather preliminary due to the lack of thorough analysis of real working traffic loads, and there might be better dynamic popular prefix loading policies/methods.

Dynamically updating route caching on routers is not new though [2,3]. Historically, routers have kept a central master routing table and the satellite processors each keep only a modest cache of recently used routes. If a route is not in a satellite processor’s cache, it will request the relevant route from the central table. At high speeds, the central table can easily become a bottleneck because the cost of retrieving a route from the central table is many times (as much as 1000 times) slower than actually processing the packet header. So the solution is to push the routing tables down into each forwarding engine. Consequently this kind of hardware cache architecture has been overwhelmed by TCAM/SRAM techniques due to the prohibitive cost of a cache miss. Lately as the RIB size has been increasing rapidly, the routing scalability problem urges us to revisit the route cache mechanism. Kim et al. [4] argue that dynamic route caching is still viable because the Internet traffic exhibits high degrees of temporal locality (as packets are grouped into flows, which are often transmitted in bursts) and spatial locality (as many hosts access a small number of popular destinations). However, they measure these features only based on a uniform class prefix (i.e. with a fixed-length of/24). Their evaluation on the least recently used/least frequently used (LRU/LFU) cache replacement strategy is also based on this uniform granularity of aggregation. Although applying the uniform class prefix avoids prefix overlapping problem on route updating, it makes the cache size even bigger than present RIB size. In their experiment, the routes are non-aggregatable, and there are approximately 500 k–1 M routes to be stored in the cache for an acceptable route miss rate. The gain of CIDR aggregation is eliminated in this method. Their research gives no description about the traffic’s locality features on the basis of CIDR prefixes as in BGP route tables.

Another related work in [5] addresses similar route cache functioning, but from the perspective of ID/Locator mapping mechanism designed in ID/Locator split schemes, and discusses the cost of route caching on the granularity of BGP prefixes. Their research focuses on the efficiency of the cached mapping information in a scenario of LISP [6] with pull-based ID/Locator mapping mechanism. In this scenario, before a packet enters the LISP transit tunnel the router has to query the mapping server on demand and caches the information (mapping of a prefix to the tunnel’s egress point) for later use. In [5], the cache policy involves a timeout threshold for all cached entries. Each cached prefix has a timer to record the time since its last hit. Once the timer runs over the timeout threshold, the prefix is expired and dumped (similar to the OpenFlow cache strategy). A cache hit will trigger a renewal of its timer and a cache miss will trigger the loading of this prefix into the cache. Actually the timeout threshold is a tuning knob to adjust the trade-off between the cache size and the number of misses. This caching policy expects popular prefixes repeatedly being hit from time to time by the working traffic, and therefore would not be easily dumped. The authors of [5] believe the cache size in LISP is much less than today’s FIB. However, their evaluation is based on a traffic trace collected on the campus network, which may not have massive subscribers like major ISPs. Moreover, the result shows at busy hours the cache size has to be unusually large. Therefore, theoretically the size of a cache has no definite upper bound.

As a good extension of traffic profiling at the level of a single IP flow (concerning individual destination IP address), some recent traffic modeling studies [21,22] pay special attention to the characteristics of conglomerated traffic over IP prefixes. In [21], the Internet prefix-level traffic behavior has been investigated and the spatial clustering feature of traffic of a tier-1 ISP has been validated. In [22], Kohler et al. investigate the structure of addresses contained in IP traffic and explore the multifractality of Internet traffic with different prefix lengths (from the scale of /0 to /32). Their measurements observe the essential dynamics of Internet traffic over a certain span of address spaces. However, their studies cannot address the interference between the traffic and real BGP prefixes. This limitation is partly due to the fact that they are using anonymized traffic traces.

To the best of our knowledge, there is no comprehensive research on the dynamics of AS level traffic loads over real BGP prefixes. We try to highlight some features of working traffic on this aspect and apply them to facilitate pragmatic route caching mechanisms.

3. Route caching and its replacement strategies

In the route caching model which has been illustrated in section 1, there is a basic tradeoff between the cache size and the miss rate. Intuitively the bigger the cache size the lower the miss rate if we apply effective cache strategies. Therefore, our goal is to minimize the cache miss rate with a given limited cache size. With this consideration, we hope that as many as possible popular prefixes will be stored in the cache, because they have a higher probability of being hit.

All above mentioned cases, no matter VA or OpenFlow or even LISP mapping caching may comply with this route caching model. The key issue is to catch the popular prefixes in their route reservoir, assumed as a RIB, without loss of generality.

---
1 The prefix blocks assigned by RIRs and assumed no overlapping, a simplification taken in this research.
2 Also comply with any other ID/Locator separation schemes.
The challenge is that the popularity of a prefix is dynamically determined by the interfering of two changing elements: the RIB entry and the working traffic loads on it. It is already known that the Internet traffic exhibits locality properties (temporal and spatial). The traffic load over prefixes has a skewed distribution [7], i.e. some are more popular than others. Route caching mechanism requires appropriate cache replacement strategies to match the fundamental dynamics of the working traffic over CIDR prefixes. We will discuss some relevant cache replacement strategies before we present our prediction based method which takes advantage of the features of working traffic.

### 3.1. Static strategy

This is the most straightforward strategy and it assumes that most of the popular prefixes are stable. The cached routes will not be updated for a long time. Normally the cached popular prefixes are chosen by hand (based on personal experience or preference). Any replacement requires administrators’ involvement. Therefore, the administrative cost can be extraordinarily high.

### 3.2. Cache-miss-based dynamic strategies

For this kind of dynamic strategy, the cache replacement can happen at any time (promptly triggered by a cache miss). There are different algorithms, but LRU and LFU are the most common strategies.

- **Least recently used (LRU).** LRU, as the name shows, will dump the least recently used prefix out when the cache is full and new routes have to be cached. This strategy stresses the temporal locality which assumes the least recently used prefix will not likely be needed in the close future.

- **Least frequently used (LFU).** LFU can be regarded as the counterpart of LRU. LFU will dump the least frequently used prefix instead, if a new prefix needs to be cached and the cache is full. This strategy stresses spatial locality, which assumes the least frequently used prefix has the lowest probability to be required in the close future.

- **Timeout dump (TD).** This strategy is proposed in [5]. Theoretically the cache size is unbound. The TD maintains a timer of not being hit for every prefix in the cache. Once the timer gets over a threshold, the prefix should be dumped. The number of cached routes scales with the volume of the timeout threshold and the traffic loads. At busy hours, the hitting prefixes may have a great diversity, which means there probably will be a lot of routes to be cached. Anyway the timeout threshold can be used to control the cache size in the worst case. Of course, one may design a variant TD by setting a cache-let-in threshold which means a prefix can only be cached when its hitting counter is higher than a let-in threshold. Therefore, the cache size can be manipulated by these two thresholds.

All above cache-miss-based algorithms are conducted on packet-level, and may require replacement when a cache miss happens. The overhead of this kind of strategies can be extremely high.

### 3.3. Optimal (Opt)

The Opt strategy is the theoretical limit as it requires future knowledge. The Opt algorithm dumps prefix that will be least needed in the future; therefore, it has the least cache miss rate. Note that the Opt can not be implemented in practice and is used as a theoretical comparison.

All these replacement strategies tend to catch popular prefixes in a route caching mechanism. Except the static strategy, the others are adaptive to the traffic dynamics. Their efficiencies depend on how good they may fit the property of the traffic load distribution across prefixes. We need to have a general analysis on these features of the Internet traffic before we can justifiably propose our purposive contribution.

### 4. Internet traffic analysis and estimation

#### 4.1. Internet traffic features

Many network measurement studies demonstrate that the Internet traffic has a feature of power law distribution [8–12], which means a few popular prefixes contribute most traffic load and the rest prefixes in the FIB have only trivial or even no traffic for a certain period of time. This feature has been observed in the past decade and hopefully will hold up in the future. Based on this understanding, we propose to sacrifice the forwarding performance of this trivial traffic in order to reduce the size of FIB. The focal point of the technique is to catch the popular prefixes.

The dynamics of the popular prefixes is complicated. Some studies simply believe the popular prefixes are rather stable. If this assumption always holds to be true, a static cache can be built and is the best strategy. However, our observation and other experimental research [7,22] show the popularity of a prefix only has short term stability. The traffic on a prefix can be very bursty, and a popular prefix can not have heavy traffic loads for all the time. The popularity of a prefix is most likely unstable when observed at Internet core routers. However, the dynamics of a traffic load over a span of address space shows a property of self-similarity. Self-similarity also manifests itself as long-range dependence (or long memory) in the time series of arrivals. This means that there are non-negligible correlations between the arrival counts in time intervals that are far apart. We suspect that the traffic on a popular prefix is not only bursty but has a memory at different time scales as well. To retrieve the memory from the traffic traces may help us to estimate the popularity of a prefix in the future. Modeling this traffic property is beyond the realm of this paper. Nevertheless, this empirical understanding is particularly helpful on designing a practical cache replacement method.
Typically, dynamic cache updates are triggered by a cache misses. With the above mentioned dynamic cache-miss-based strategies, such as LRU/LFU, or TD, some bursty traffic on non-popular prefixes might trigger unnecessary replacements. Therefore, probably real popular prefixes have to be swapped frequently and will incur more misses consequently. If non-popular prefixes burst from time to time, the popular prefixes will be extremely slippery to be caught.

In order to minimize the misjudgment incurred by bursty traffic, the popularity of a prefix should be retrieved from its traffic history. More specifically, if we want to catch the popular prefixes precisely and hold them in the cache as long as possible, it needs to refer to some range of traffic trace instead of its instant traffic load.

4.2. Impact of sampling measurement on traffic estimation

Another consideration of the traffic estimation is the sampling ratio of traffic measurement. For ASBRs, to record every packet of the traffic may result in considerable burden on router's/line card CPU. According to Cisco's technical white paper [23], When enabling NetFlow on the Cisco 12000 Series Router engine 0 and engine 1 line cards, the hardware forwarding mechanism is bypassed, and packet-forwarding decisions are made in software. Cisco 12000 Engine 2, 3, 4" and 5 line cards have NetFlow implemented on a hardware Application-Specific Integrated Circuit (ASIC), so they can switch NetFlow packets at or near line rate. In addition, because all this occurs in hardware, there is no performance penalty on the CPU of the line card. However, the high packet forwarding rates and high flow counts on the Cisco 12000 can generate a large amount of NetFlow traffic. The line card CPU is then responsible for processing and exporting the flows. Therefore, when using a software-based NetFlow implementation on the Cisco 12000 Series Router, sampled NetFlow is recommended. The technical report [23] shows that a sampling ratio of 1:100 will have average 75% decrease in the CPU utilization and a ratio of 1:1000 will have average 82% decrease respectively. Cisco 12000 is used as an example to illustrate the consideration of sampling ratio on collecting traffic traces. This consideration is valid for any model and even brand. Setting sampling ratio is a common configuration for traffic measurement on core routers.

Nevertheless, sampling trace will introduce estimation errors when reversing the traffic load from a sampled trace. For example, if the traffic over a prefix is not sampled, there will be no estimated traffic of this prefix. Or if a trivial traffic happened to be sampled, the estimated traffic load may be much higher than reality since we are going to inverse the traffic load by multiplying the sampled counts with the sampling ratio. In order to reduce the estimation error, according to centric limit theorem, we need to increase the number of samples. In our case, to maintain the same sampling ratio, a longer history of traffic trace should be referred when estimation the traffic on each prefix.

Setting the range of traffic history is vital to both the validity of our traffic estimation and the corresponding overhead in our method. First of all, the range of the traffic trace should be long enough to guarantee the preciseness of the traffic estimation. On the other hand, since the trend of traffic load may vary over a long period of time, the range of history can not be arbitrarily long. Moreover, too much history record may incur undesirable overhead. Our experiment gives a reasonable range of traffic traces for the traffic prediction which will be illustrated in Section 6. We also evaluate the impact of traffic sampling on our popularity prediction method in Section 6.

4.3. Grey model for traffic prediction

4.3.1. Grey model for traffic prediction

On predicting the traffic over different prefixes for a certain period of time, we apply regression analysis in statistics since we can hardly find any explicit model to appropriately depict the relationship between traffic loads and their influential elements. Suppose we have collected a series of traffic load records for each prefix within regular time intervals. Our regression is to estimate the parameters of the regression function. Then this function can be used to predict the traffic load at the next time interval.

However, there are some difficulties when we apply polynomial regression to model the relationship between traffic load and the time. Firstly, it is difficult to set the order of polynomial to fit all prefixes well. If the order of the polynomial is too low (linear e.g.), there must be too much residual. On the other hand, higher order may require too many history records to guarantee the preciseness of prediction. For sampled traces, it is difficult to collect enough valid information.

To get around the above mentioned difficulties, we apply the grey model [19], which is typically used in grey system control and prediction. A grey system is between white system and black system. For a white system, we know fair enough details of its organization and function. For a black system, we have almost no useful information about its mechanism. In most cases, when we have some information of a system but lack enough data to precisely depict the system, the grey modeling is an exceptionally useful technique.

In our application, we are going to estimate the popularity of each prefix according to sampled traffic trace. We use a grey model to depict the dynamics of the popularity. The grey model assumes that although the elements that influence the traffic loads cannot be depicted with explicit functions, their general effect has been illustrated in the traffic traces. Modeling the traffic load as a grey system may help us estimate its overall behavior in the next stage. This model can effectively retrieve useful information from limited history data. The grey model in our application studies traffic load which is the single variable. We model the cumulative of this variable to comply with a first-order differential equation. This model is literally GM(1,1).

The process of GM(1,1) prediction can be summarized as follows: for a series of historical records of a variable, a cumulative series needs to be generated at first. Then a first-order differential equation of the cumulative is used to model the general trend of development of the variable. The parameters of the equation can be estimated through least square method. After that, on substituting the estimated parameters in the differential equation, we can have
the estimated cumulative value for the next period of time. Finally, inverse the cumulative process, we have the estimated value of the variable for the next interval.

The mathematical analysis of the grey model can be found in [19]. In this paper, we only deliver the GM(1,1) modeling process in our prediction application.

On applying GM(1,1) model, the variable is the traffic load of a prefix for a certain time interval. By collecting the traffic loads of each prefix within every time slot, we have a series of traffic loads for each prefix as \( X(t) \), where \( t = 0, 1, 2, \ldots \); after an accumulation generation operation (AGO), we have \( X^{(1)}(t) \), the cumulative series of \( X(t) \), and formally

\[
X^{(1)}(t) = \sum_{i=0}^{t} X(i), \quad t = 0, 1, 2, \ldots
\]

The GM(1,1) model mathematically assumes the cumulative series is continuous and abides with the following differential equation:

\[
\frac{dX^{(1)}(t)}{dt} + aX^{(1)}(t) = u. \tag{1}
\]

After Laplace transform and inverse Laplace transform, we have the solution of the differential equation:

\[
X^{(1)}(t+1) = \left( X^{(1)}(0) - \frac{u}{a} \right) e^{-at} + \frac{u}{a}. \tag{2}
\]

In Eqs. (1) and (2), \( a \) and \( u \) are parameters that can be estimated through least square method (\( \hat{a} \) and \( \hat{u} \) are the estimation of \( a \) and \( u \) respectively).

\[
\hat{A} = (\hat{a}, \hat{u})^T = (B^T B)^{-1}B^T Y, \tag{3}
\]

where

\[
B = \begin{bmatrix}
-0.5(x^{(1)}(2) + x^{(1)}(1)) & 1 \\
-0.5(x^{(1)}(3) + x^{(1)}(2)) & 1 \\
\vdots & \vdots \\
-0.5(x^{(1)}(t) + x^{(1)}(t-1)) & 1
\end{bmatrix},
\]

\[
Y = \begin{bmatrix}
x^{(1)}(2) \\
x^{(1)}(3) \\
\vdots \\
-x^{(1)}(t)
\end{bmatrix}.
\]

After that, we substitute \( \hat{a} \) and \( \hat{u} \) into Eq. (2), and then inverse the AGO operation. The prediction of the traffic load will be

\[
\hat{X}^{(1)}(t+1) = \hat{X}^{(1)}(t+1) - \hat{X}^{(1)}(t).
\]

This model treats the accumulative traffic load as the single variable that its current variance is only subjected to its previous amount. This model establishes a first-order differential equation shown in (1). The GM(1,1) model, namely a single variable \( X^{(1)}(t) \) and its first order differential equation, has been widely used in grey prediction applications.

Technically, if \( X^{(1)}(t) \) is high, we hope its variance accordingly low, which means a stable popular prefix \( X^{(1)}(t) \) (has a high volume with low variance) matches the model better than otherwise. Those prefixes that do not meet the condition are not predictable. According to the grey modeling theory, the parameter, \( a \), depicts the development of \( X^{(1)}(t) \). It has a forbidden area: \((-\infty, -2] \cup [2, \infty)\). The proof can be referred to [19]. Once the estimated parameter \( a \) falls into a forbidden area, the prefix is not predictable and cannot be a candidate stable popular prefix. In this case, the traffic over such a prefix is random. We believe there is no useful information that could be extracted from its traffic trace. In our algorithm, we only accept the prediction with \( a \in (-2, 2) \), since it does not make much sense to predict the unpredictable prefixes. This method can extract useful information from a grey system efficiently.

The grey model has the following virtues. Firstly, the accumulative series depicts a general trend of traffic with a certain degree of confidence, which is essential to our traffic prediction. Secondly, it is adaptive to bursts of the traffic loads over prefixes. On the contrary if we estimate the popularity of a prefix by simply calculating an averaged traffic load over a period of time, the traffic burst of a non-popular prefix may influence our judgment. Thirdly, the unpredictable prefixes can be singled out easily (once the parameter \( a \) falls into the forbidden area). Lastly, the grey model requires not too much input information compared with other models. It is particularly suitable to the situation of sketchy forecasting.

5. Prediction based cache strategy and its implementation

Based on the GM(1,1) traffic prediction, our route caching strategy will dump non-popular routes out and load potential popular prefixes periodically.

In our scheme, we just need some sampled traffic traces to guide our prediction. Instead of triggering a cache replacement by a cache miss, we can estimate the popular prefixes of the next stage beforehand. The cache loading/dumping is solely based on this prediction and takes place with certain intervals. The size of the cache and the intervals of the cache replacement may vary according to the hardware capacity.

In our strategy, at first we need to set a replacement interval (such as 5 min, 1 h or even longer). This setting is a tradeoff between performance and overhead. A shorter interval may have higher cache efficiency but also introduces more computational overheads. A router with light routine computational burden but less resource of high performance memory tends to set shorter intervals. On the other hand, a router with less powerful CPU but relatively more memory may set longer intervals.

In order to reduce the overheads of frequent cache/route replacement, we suggest using two independent memories for forwarding lookups. And the forwarding lookups will be carried on in turns with these two memories. When one memory is used as a working cache, the other is used for loading the caching routes on the next stage. As soon as the replacement happens the forwarding lookups will be switched to the alternate memory. This mechanism guarantees there will be almost no delay of...
cache replacement, which is extremely important for high speed routers. If considering the popular prefixes being cached only accounts for approximately 5% of the RIB, the backup mechanism only consumes 10% of the RIB size altogether. Our method successfully reduces the memory requirement in the forwarding devices an order of magnitude.

In the case of route updates, the FIB loading can be triggered immediately without waiting for the next due time of a cache replacement. There is a copy of the current route cached in the FIB, therefore any route update that involves the current cached routes can be updated promptly. After being updated, the standby memory can be activated as the working cache, which means a cache memory switch can be promptly triggered by route updates.

Another issue that needs to be taken into consideration is the inconsistency of loading overlapped prefixes. In order to guarantee correct forwarding with longest prefix match algorithm, when a less specific prefix is loaded as popular prefix, its more specific sub-prefixes have to be loaded too. For example, if we have two prefixes in the RIB: “10.10.0.0/16” and its sub-prefix “10.10.10.0/24”, when a traffic flow has a destination of “10.10.8.8”, all packets in the flow will be counted as hitting traffic on “10.10.0.0/16”. In this case, “10.10.0.0/16” is popular but “10.10.10.0/24” is not. However, when we load “10.10.0.0/16” into the cache, “10.10.10.0/24” should be loaded too (unless the two prefixes have the same next hop).

Otherwise, there will be a routing problem. These overlapped prefixes make the cache size bigger than the popular prefix set that we estimate. Sometimes a prefix may have as many as more than hundreds of sub-prefixes. In this case we optionally adapt a simple FIB suppressing algorithm, Optimal Routing Table Constructor (ORTC) [20], to reduce the number of sub-prefixes and consequently reduce the ratio of the cache size and the accommodated popular prefix set.

The basic algorithm of ORTC is firstly to put every overlapped prefix into a tree structure as nodes. The root is the least specific (covering) prefix. The ORTC algorithm may generate new nodes in the tree while guarantees the forwarding behavior unchanged. Any nodes with the same next hop with its closest parent node can be eliminated from the tree. After ORTC operating, the nodes close to the root of the tree will have the most prevalent next hops. Nodes that are located near the leaves of the tree should have less prevalent next hops. Finally, this algorithm should be applied recursively, within every sub-tree.

In the ORTC algorithm, we define the operation $A\#B$ on two sets of next hops:

$$A\#B = \begin{cases} A \cap B & \text{if } A \cap B \neq \emptyset, \\ A \cup B & \text{if } A \cap B = \emptyset. \end{cases}$$

We define the function $\text{Inherited}(N)$ on nodes other than the root:

$$\text{Inherited}(N) = \begin{cases} \text{nexthops(parent}(N)) & \text{if } N \neq \emptyset, \\ \text{inherited(parent}(N)) & \text{otherwise}. \end{cases}$$

There are three steps in the algorithm of ORTC.

### ORTC algorithm

**Step One:** make sure each node has either two or none child.

for each node $N$ (root to leaves) {
   if $N$ has exactly one child node, 
      create the missing child node;
   if nexthops($N$) = $\emptyset$, 
      nexthops($N$) = inherited($N$);
}

**Step Two:** pass prevalent nexthops from leaves to root.

for each node $N$ (leaves to root) 
   {If $N$ is a parent node, 
      nexthops($N$) = nexthops(left($N$))\#nexthops(right($N$));
   }

**Step Three:** eliminate nodes which have the same next hop with its closest parent.

for each node $N$ (root to leaves) 
   {if $N$ is not the root and 
      inherited($N$) \in nexthops($N$) 
      nexthops($N$) = $\emptyset$;
   else 
      nexthops($N$) = choose(nexthops($N$));
}

The complexity of the ORTC algorithm is linear to the number of nodes in the tree. The number of tree nodes is $O(wN)$ where $w$ is the maximum number of bits in the prefixes and $N$ is the number of prefixes in the input routing table. In IPv4, the BGP prefixes have the maximum length of 24 bits, and the shortest prefixes have a mask of 8 bits. Therefore, the maximum height of the tree is $24 - 8 = 16$. $N$ is proportional to the number of overlapped prefixes.

In our scheme, we only apply ORTC FIB compressing when a popular prefix has sub-prefixes in the RIB. We set the popular prefix as the root of the ORTC tree. Its sub-prefixes are nodes and leaves. The complexity of ORTC in our scenario is $O(16mn)$, where $m$ is the number of top level popular prefixes, and $n$ is the average number of sub-prefixes for them. Therefore the complexity can be much less than applying ORTC on the whole route table.

It has been proven in [20] that ORTC is the optimal FIB suppressing algorithm. The gain of ORTC depends on the number of forwarding next hops and the forwarding redundancy of routes (which is related to connectivity/topology and the number of interfaces on the router). Our simulation shows that ORTC can effectively reduce the number of sub-prefixes for most popular prefixes. In case the FIB suppression does not have much gain, the ORTC process can be omitted.

The only limitation of ORTC is the problem of route update. Because the next hop of a prefix may be different with the corresponding entry in the RIB and even new prefixes can be generated, each route update requires recalculation of the entire tree. The incremental route update is not possible in this case. However, in our scheme the recalculation will be limited to relevant popular prefixes. Moreover, a standby memory is always ready for route updates and popular prefix swaps. Generally the overheads of ORTC recalculation and route updates are acceptable.
Our method is different to the cache-miss-based replacement strategies. First of all, it is prediction-based. The cache replacement happens at discrete time slots (periodically). All popular prefixes swap is biased by the beforehand estimation, and stable popular prefixes are not easy to be dumped out. On the contrary, cache-miss-based methods may dump out a real popular prefix on every cache miss. LRU/LFU and TD may loss the few triggering packets of a flow before loading a popular prefix into the cache. When the dynamic is high (a lot of swap) there will be a high miss rate. In most cases, a bursty traffic will incur frequent cache replacements according to our experience.

Secondly, our method only considers the general traffic load of a prefix since the cumulated traffic load is the single object in the GM(1,1) model. Therefore, our method effectively alleviates the impacts imposed by traffic bursts or sampling errors. Some bursty traffic may hit a certain prefix for a while, but normally the bursts will not last long. In other words, the burst tends to be short and unpredictable. If we promptly swap cached items, the short bursts may incur more cache misses.

Lastly, our method gives a possible solution to the prefix overlapping problem which is one of the major challenges in the route caching mechanism. Because the prefixes are sometimes overlapped with each other, loading a popular prefix may require loading more than one prefixes at a time. LRU and LFU need to decide which prefixes should be swapped to make room for the newly entering prefixes. This decision can be difficult and complicated since there are possibly no more prefixes which is less frequently used or less recently used than the entering prefixes. For example, prefix A needs to be cached together with its three sub-prefixes. There is only one suitable prefix that can be dumped out from the cache. It is complicated to evaluate the loss and gain of different decisions in their algorithms. The frequent cache misses will trigger such evaluation and incur prohibitive computational overheads. Therefore, LRU and LFU prefer to use uniform class prefixes in order to avoid prefix overlapping (can be devised as in [4]: uniformly/24). However, this resort devastates the aggregation of IP prefix and introduces too many fragmental prefixes. In our method, we take the prefix overlapping into consideration. In addition to adapting ORTC algorithm to suppress the sub-prefixes, we set cache replacement in a batch process. On ranking the popularity of each prefix through traffic load prediction, the popular prefix set can be adaptive to the cache size conveniently.

The overall implementation of our scheme can be illustrated in a conceptual router model as shown in Fig. 1.

We evaluate our method with the working traffic traces collected from the China Telecom backbone networks and CERNET core routers. We also study the impact of sampling traces, as well as the overhead of our method, via simulation.

6. Measurements and evaluation

6.1. Methodology

Our simulation uses working traffic traces and corresponding route tables provided by CHINANET [14], the backbone of China Telecom. CHINANET reaches more subscribers in Asia than any other IP networks and provides direct connections to all major global ISPs. CHINANET has more than 47.18 million registered customers by the end of 2008 and provides daily access for more than 28 million people.

The traffic traces provided by CHINANET are sampled flow-level NetFlow traces in a period of two months (from 14th November 2009 to 16th January 2010) at different backbone routers located on two PoPs of CHINANET. A flow is defined as a 5-tuple (i.e., source IP/port, destination IP/port, protocol). In CHINANET, China Telecom IP network services are delivered over the international 40 Gbps MPLS-enabled IP backbone. Since the volume of the traffic transited by those routers is extremely large, the NetFlow
sampling rate has been set as 1:5000 with the sampling techniques specified in [13]. Basically our experiments are based on this sampling ratio.

We choose traffic traces from two core routers in Shanghai and Guangzhou respectively to generate per-prefix traffic statistics. We find that a small fraction of Internet prefixes carried a large majority of ISP traffic within a short time span. This observation is also found in previous studies [1,12,15–17]. Our measurements confirm this property.

In order to analyze the impact of sampling, we also collect packet-level traffic data on the gateway of CERNET as well as the data provided by CAIDA. Considering that the data from CAIDA has been processed with IP anonymity techniques, we generate a route table with uniform/18 prefixes that cover all IP address space. In this way, each anonymized IP packet will hit one prefix in this route table. Since our simulation on these data is focused on the sampling impact, the total number of popular prefixes is not our concern in this case. We simulate the situations with and without traffic sampling on guiding the cache strategies. Our simulation with these un-sampled traffic traces in pcap format gives us a basic understanding of the impact of sampling. It is worth noting that traffic sampling may incur errors on estimating the volume of traffic: some prefixes have no traffic when being estimated with the sampled trace; on the other hand, the traffic over some other prefixes can be overestimated. According to the central limit theorem of random samples, we have to increase the number of samples to reduce the estimation error. Specifically, in our method, at a given sampling ratio, the range of referring traffic traces needs to be prolonged as long as the traffic trend may not change substantially during the referring time span. The simulation shows our method is not sensitive to the sampling ratio when compared with other cache strategies, since the grey model is suitable to retrieve useful information from limited data sets. In Section 6.2, we will illustrate the impact of sampling ratio on different cache strategies. If we set a proper range of referring traffic traces, our method outperforms the other strategies on predicting the popularity of each prefix.

As complementary data to the major ISPs, the traffic in the CERNET networks represents most stub ASes at the edge of the Internet. However, our analysis will focus on the dataset collected from CHINANET, and the CERNET traffic has the similar outcome in our simulation.

In our method, not every prefix has the same predictability. When the traffic trace of a prefix reveals reliable trend instead of randomness, the prediction is meaningful. On evaluating the reliability of popular prefixes prediction, we apply a no unit metric commonly used in signal processing (defined as mean/stdev), which is an alternative form of Signal Noise Ratio (SNR) and exactly a reciprocal of the coefficient of variation. In this definition, \( SNR = \mu / \sigma \), where \( \mu \) is the signal mean or expected value and \( \sigma \) is the standard deviation of noise, or an estimate thereof. Notice that such an alternative definition of SNR is only useful for variables that are always positive. Sometime SNR is defined as the square of the above definition. The prefix with relatively high SNR means the trace is informative enough to be predicted with future traffic loads. On measuring this metric of each prefix, we can estimate the appropriate size of a predictable popular prefix set at different cache update time scales. In Section 6.2, we will discuss the meaning of this metric and the result of our experiment.

At last we evaluate different cache replacement strategies over sampled working traffic traces (from 14th to 16th of January 2010) at given cache sizes. A reasonable range of reference history has been experimented on these working traffic traces. Our experiments give an optimal range of time span for maximizing the number of samples with limited influence of traffic trend variance.

The following specifications of our measurement will give some clarification of our method.

Firstly, the traffic load is measured in terms of the number of packets instead of Bytes, because our study is focused on routing performance instead of bandwidth utility research.

Secondly, since the BGP prefixes can be overlapped, as we calculate the popularity of a prefix, the hitting traffic accounts only for the longest prefix matching. As we have addressed previously, the size of the popular prefix set is proportional to the final cache size. Although our method can optionally apply ORTC or other practical FIB suppressing technologies, on analyzing the performance of route caching mechanism we simply use the raw size of the popular prefix set as a metric. We assume other cache strategy may not apply the same FIB suppressing technology, because cache-miss-based strategies cannot promptly recalculate the ORTC tree on each cache miss.

Thirdly, on evaluating our method together with other related cache strategies, we measure the rate of cache miss by simulation. We replay the traffic trace over the popular prefix set loaded in the route cache. Any packet that cannot match any prefix in the cache, will be counted as a miss. Actually the miss rate is related to the diversity of the destination IP addresses and the given cache size. For cache miss based strategies, we start to calculate the cache miss rate when the cache is full. In other words, we cut out the initial stage on calculating the average cache miss rate of each strategy. In Section 6.3, we will show the overall performance of a cache miss based strategy has nothing to do with initially preloaded cache items.

Finally, on simulating TD, we set different timeout values for different cache sizes. Basically the number of cached prefixes depends on the timeout value. We need to tune the timeout value according to the given cache size. Table 1 gives some timeout values with corresponding cache sizes. In order to avoid cache overflow when TD has no obsolete entry to be dumped, we choose not to load any new prefix on that cache miss. This is the main difference to LRU.

6.2. Observations

We have the following observations in our simulation. These findings give a profile of the working traffic.

Firstly, the working traffic load over BGP prefixes has a power law distribution. Fig. 2 shows our observation of the traffic distribution across BGP prefixes on 14th JAN 2010. Of all the 336,390 prefixes, we calculate the number of packets hitting on each prefix. Our observation is that
143,950 prefixes (account for 42.8% of RIB) have no traffic at all that day (partially because of the sampling, un-sampled trace may hit more prefixes with trivial traffic). And when we sort the rest prefixes by their daily traffic loads in terms of packets and plot it in a log–log format, we get a linear track for the most popular prefixes, which is typically a Zipf distribution. The slope of the fitting line is \( -0.607 \). The rest part of the track deviates from the fitting line. Anyway, the distribution of traffic load on each prefix is skewed. It can be seen clearer in a CDF format of the line. When we sort the rest prefixes by their daily traffic loads in terms of packets and plot it in a log–log format, we get a linear track for the most popular prefixes, which is generally should be higher than otherwise. We plot the traffic load on each prefix for every 5 min in a day, and consequently we have a discrete series \( x \) representing the traffic loads at 288 discrete time slots of a day for each prefix. In this \( N \times 288 \) matrix (288 × 5 min in 24 h, \( N \) is the number of RIB entries), we find the traffic loads on a given prefix at successive time slots may change drastically from time to time.

Secondly, not every prefix is equally predictable in terms of its popularity according to its traffic trace. In order to illustrate the dynamics of the variable popularity, we calculate the traffic on a smaller time scale (at an interval of 5 min). That is to say we count the packets on each prefix for every 5 min in a day, and consequently we have a discrete series \( x \) representing the traffic loads at 288 discrete time slots of a day for each prefix. In this \( N \times 288 \) matrix (288 × 5 min in 24 h, \( N \) is the number of RIB entries), we find the traffic loads on a given prefix at successive time slots may change drastically from time to time.

On observing the stability coupled with its popularity of a prefix, we calculate the mean value and variance of \( x \). Then we give a generalized metric to measure the popularity and stability of a prefix simultaneously: the ratio of mean/stdev as shown in formula (4), which is the SNR of variable \( x \)

\[
SNR = \begin{cases} 
\frac{\mu}{\sigma} & \text{if } \mu > 0, \\
0 & \text{if } \mu = 0.
\end{cases}
\]

In which,

\[
\mu = \frac{1}{288} \sum_{i=1}^{288} x_i; \quad \sigma = \sqrt{\frac{1}{288} \sum_{i=1}^{288} (x_i - \mu)^2}.
\]

Obviously, a stable popular prefix will have substantially high traffic loads and relatively low variance, and its SNR generally should be higher than otherwise. We plot SNR value of each prefix (sorted by ranks) in Fig. 4. A small portion (2–3% of RIB size) of prefixes (from point A to B) has high SNR value. These prefixes are relatively stable and contribute significant traffic loads and should be kept in the cache as long as possible. Although not confirmed, we suspect this portion of stable popular prefixes is the popular prefix set measured in [1] which accounts for about 2% of the RIB. Other prefixes (from point B to C) have changing popularities, and their traffic traces are still informative. However, their SNRs decrease, which means their traffic traces become less informative. Beyond point C, they are unpredictable prefixes since their SNRs are extremely low. Fortunately those prefixes are mostly non-popular ones. The above observation indicates a portion of the BGP prefixes that their traffic loads are predictable. Empirically, based on this understanding, in our method we set the range of estimated parameter \( \hat{\alpha} \in (-2, 2) \) of the GM prediction as acceptable range. Otherwise, for \( \hat{\alpha} \not\in (-2, 2) \), the prefix is unpredictable and the prediction does not make much sense.

The third observation is that the impact of packet sampling can be negligible in our method. Considering the volume of traffic on core routers, sampling of traffic can effectively reduce the overhead of data collection and processing. As we have mentioned in section 4.2, the reduction of CPU utilization is dramatic when NetFlow packet sampling is implemented. However, a sampled trace may incur estimation error of the traffic volume, and the error depends on the number of samples. According to the central limit theorem, we can reduce the error by...
increasing the number of samples. At a given sampling ratio, our prediction should refer to sampled traces over a prolonged time span. To be more specific, if we want to predict the traffic of the next 5 minutes, we refer to the traces in the previous a dozen of 5 min. Theoretically, packet sampling introduces effects of low-pass filtering, which eliminate high frequency components of signals and may smooth the peaks of bursts sometimes. But sampling will not change the power law distribution of the traffic loads over BGP prefixes. Generally packet sampling will not influence our estimation of the most popular prefixes. On analyzing the impact of packet sampling, we use both sampled and un-sampled traces from CAIDA and CERNET. We find that our method will not be much sensitive to the sampling ratio we set a proper range of referring traces. When applying our method on the sampled traces, most popular prefixes can be caught effectively even at a low sampling ratio of 1:5000.

For this analysis, we use publicly available trace from CAIDA [24], which was collected from an OC-192 link consisting of 2,127,512,272 packets covering an hour time window, and another trace from CERNET collected from an OC–192 link consisting of 1,346,872,338 packets covering one hour. We set a route table of 262,144 uniform/18 prefixes (that cover the whole IP space) for the anonymized traces from CAIDA, where/18 is used to mimic the prefix coverage in BGP. We use real BGP route table collected from CERNET routers for CERNET trace instead. In our simulation, we use the previous 10 × 5 min traces to predict the popular prefix in the next 5 min. Considering that the route tables have different size, we set different cache sizes accordingly. The cache miss rate is calculated according to the real trace. Table 2 shows the difference of estimation of the most popular prefixes guided by sampled and un-sampled traces. The difference of cache miss rate on applying sampled and un-sampled traces is negligible.

The last observation is the marginal benefit of the cache size on working traffic trace. Theoretically the cache size is proportional to the cache hitting rate when applying effective cache strategies. However, there is a marginal benefit when the popularities of different prefixes have a fat-tail power law distribution. It means beyond certain cache size, further increasing the cache size will not have much improvement of the cache hitting rate. We apply optimal strategy to show the theoretical limits at different cache size and replacement frequencies. This strategy can preload popular prefixes accurately since it had the future information. These theoretical limits can be different since it solely depends on the features of the real traffic.

On plotting the limits of different replacement frequencies and cache sizes in Fig. 5, we get some basic understandings of the Internet traffic. Firstly, at a given cache miss rate and replacement interval, we can estimate the minimum cache size which is the vital parameter in our method. Second, as we can see in Fig. 5, there is a diminishing marginal utility of caching more prefixes: soon after the size of the cache exceeded 5% of the RIB size (about 18,000 entries), the miss rate decreases slowly with the increasing size. After loading 10% of the RIB (33,000

<table>
<thead>
<tr>
<th>Traces</th>
<th># of prefixes that contribute traffic</th>
<th># of common popular prefixes</th>
<th># of different popular prefixes</th>
<th>Cache miss rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAIDA Sampled (1:5000)</td>
<td>6660</td>
<td>4679</td>
<td>321</td>
<td>3.36%</td>
</tr>
<tr>
<td>Special route table with 262,144 uniform/18 entries</td>
<td>Un-sampled</td>
<td>20,552</td>
<td></td>
<td>3.28%</td>
</tr>
<tr>
<td>Cache update interval: 5 min</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CERNET Sampled (1:1000)</td>
<td>11,346</td>
<td>7449</td>
<td>551</td>
<td>4.97%</td>
</tr>
<tr>
<td>Real BGP route table with 334,232 entries</td>
<td>Un-sampled</td>
<td>31,865</td>
<td></td>
<td>4.91%</td>
</tr>
<tr>
<td>Cache size: 8000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cache update interval: 5 min</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
prefixes) as popular prefixes, the miss rate decreases rather slowly and almost has no obvious gains even though the replacement takes place frequently. This fact implies that numerous prefixes contribute trivial traffic loads and suggests an efficient cache size will be around the turning points. Based on this observation we can tune a better tradeoff between the cache size, cache miss rate and cache update intervals (overheads). Our method attempts to find a proper point within the tradeoff triangle.

6.3. Performance evaluation

On evaluating the performance of our method, we simulate all strategies at various cache sizes with different traffic traces. We plot average cache miss rate in Fig. 6(a)–(c). We start to calculate the average cache miss rate from the 11th 5-min time slot of a day (refer to Fig. 7a), since then our method has collected enough history data and cache miss based strategies have a full cache. Generally we have the following observations.

Static cache has the highest miss rate. The popular prefixes set we used in a static method was measured on a given day with the optimal strategy at the same router. This strategy is used in [1]. In static strategy, the most popular prefixes of that day are in the cache and will not be replaced for a long time. To some extent, the static strategy can be viewed as the extreme situation where we set the replacement interval as infinity. In fact, short-term popular prefixes can only be caught effectively by dynamic cache strategies. However, when the cache size is extremely small, it performs effectively. In most cases, the cache size is not too constrained, and the static strategy gains less than the other dynamic cache strategies. Another critical fact is that the static strategy is highly sensitive to the sampling ratio of the trace. When we use 1:5000 sampled traces to estimate the popularity of each prefix, the performance degrades drastically, because the popularity of some prefixes can be overestimated.

Cache-miss-based strategies (such as LRU/LFU/TD) can catch some of the short-term popular prefixes, and generally better than static strategy. Given a constraint cache size, the performance of TD is similar to that of LRU. However, it is difficult to set a proper dump threshold for TD to meet requirements of cache sizes. If we set the dump threshold too high, too many prefixes will stay in the cache and the cache may be at a risk of memory overflow. In most cases, a prudent configuration of the threshold will make TD inefficient. Our simulation confirms the claim in [4] that LRU has better performance than LFU. However, in LRU some bursts will influence the route caching efficiency from time to time and might incur much higher overheads.

Our prediction-based method will replace the cached prefixes periodically with determined intervals. Considering the computational overhead of a router, we choose 5 min as our cache update interval in the simulation. It can catch most of the popular prefixes solely guided by traffic traces over a span of time. It is almost immune to the impact of bursts and not sensitive to packet sampling. This method can be adaptive to both stable and instable popular prefixes at various time scales. The result of the simulation shows our method outperforms the other strategies and is remarkably close to the optimal strategy (theoretical limits).

We also present how the cache miss rate develops as time is passing. We start to calculate the cache miss rate every 5 min when the cache is full. In Fig. 7, we plot our simulation on ChinaNet traffic trace with a fixed cache size (i.e. 30,000 popular prefixes). Basically our method outperforms the other strategies at all time. We simulate the cache miss based strategies starting with an empty cache and optimal prefixes respectively. The result shows the cache miss rate will reach its average level soon after the initial stage. Even when we preload optimal prefixes into the cache at the initial stage, the cache miss based strategies may dump out popular prefixes on each cache miss, which causes more cache miss afterwards. With cache miss based strategies, some prefixes can be dumped and reloaded many times within 5 min. Unless the cache-miss-based strategy knows which prefix should be dumped, inefficient updates cannot be avoided. On the other hand, even though a cache miss based strategy can be guided with training datasets, it may have much higher update overhead than our method if cache miss happens frequently. It would be practical to update the cache items periodically.

The last notable measurement of our study is the overlapping sub-prefixes of the popular prefixes. Our previous analyses are based on the sheer popular prefix set. However, when we load popular prefixes all its covered sub-prefixes must be loaded too. In BGP, the prefix overlapping is common, and the required cache size is normally bigger than the popular prefix set. In order to accommodate as many as possible popular prefixes in a limited cache size, we propose optionally applying ORTC algorithm to suppress the sub-prefixes of each popular prefix. As plotted in Fig. 8, without ORTC, in the worst case, the cache size can be a few times as big as the sheer popular prefix set, which means we have to load many non-popular sub-prefixes into the cache. With ORTC, the number of sub-prefixes reduces considerably, and the overall cache size is about a double size of the sheer popular prefix set which
Fig. 6. Comparison of different cache strategies: (a) evaluation with sampled ChinaNet traces, (b) evaluation with un-sampled CERNET traces and (c) evaluation with un-sampled anonymized CAIDA traces.
is affordable when the popular prefix set is only 2–5% of the RIB. Hopefully we can reduce the requirement of the FIB size an order of magnitude with no more than 5% cache miss rate. Anyway the FIB suppressing ratio depends on the topology/interfaces and RIB structure. We believe many other FIB aggregation techniques can be applied optionally. The selection of candidate FIB aggregation technology is not the focal point in this paper, but our prediction-based method and FIB implementation is compatible with any other FIB suppression technologies.

6.4. Overheads

On evaluating the overheads of our method, the most concerned issues are the cost of traffic trace collecting, the calculation complexity of GM(1,1) and the delay of ORTC FIB suppressing and cache updating. The evaluation is based on analysis and experiments. According to the evaluation, the overhead of our method is acceptable on most ASBRs.

The first pertaining overhead is the collecting of traffic trace. We propose to apply packet sampling, which will reduce the overhead considerably. Since our method is not sensitive to the packet sampling, at core routers we set the sampling ratio as 1:5000. As we have mentioned in section 4.2, this sampling ratio will considerably reduce the burden of linecard CPU. We believe the other tracing tools may gain similar benefit from packet sampling configuration.

Theoretically, assuming the traffic trace has \( m \) records for each prefix, and \( N \) prefixes in the RIB, we need to store the \( m \times N \) matrix in order to conduct the grey model prediction. Suppose each prefix consumes 5 Bytes (for IPv4, 4 Bytes of IP address, 1 Byte of the prefix length) and each record takes 2 Bytes (maximum \( 2^{16} \) packets will be good enough for sampled records, otherwise for non-sampled records when the packet number exceeds 65,536 we can use the unit of Kilo packets), the memory we need to accommodate the traffic trace is \( O((5 + 2m)N) \) Bytes. We have experimented different reference ranges of traffic trace in our method (as shown in Fig. 9) and find that the optimal range of reference traffic history is about 10 update intervals (50 min). If the reference range exceeds the optimal point, there will be no more improvement on the accuracy of traffic prediction. In this case, \( m \) will be no
more than 10 with the cache replacement interval of 5 min. In our simulation program, the size of the traffic trace matrix takes no more than 10 MB memory.

On calculating the traffic load over each prefix at different intervals, we apply the patricia trie algorithm; therefore, the theoretical time complexity is $O(kh)$, while $k$ is the number of sampled packets, and $h$ is the height of the trie tree, normally no more than 16 for IPv4 BGP prefixes.

As the grey model GM(1,1) is not designed for precise prediction but for mining a general trend of a process, it is not sensitive to sampled records. Basically $k$ is much less than the real packets forwarded by the router. In our simulation, the NetFlow record has a sampling ratio of 1/5000 as it is collected from the core router at a 40 Gbps interface card. There are about 0.5 million of sampled packets for each 5-min interval at busy hours. The trace collection will be fulfilled on the line card promptly. The overhead of trace collection pertains to the packet sampling ratio. Our experiment shows the 1:5000 sampling ratio is quite acceptable for most ASBRs.

Secondly, the overhead of popular prefix prediction can be well affordable if we optimize our method with incremental update processing. According to our experiment on the real traffic trace, most BGP prefixes have no traffic during the 5 min time window. We focus our prediction on the prefixes that may carry traffic in the next time slot. Suppose the cache size is $k$ entries. Initially we load top $k$ most popular prefixes into the cache. Later on, we can update popular prefixes periodically via incremental update processing. We notice that a majority of popular prefixes may not be dumped out at each cache update. It is not necessary to predict every popular prefix at the time of cache update. An incremental update processing is good enough. We sort the popularity of prefixes according to the latest traffic trace and check the top $k$ prefixes purposefully. For the previous popular prefixes in the cache that does not rank in the top $k$ prefixes and previous non-popular prefixes not in the cache but rank in the top $k$ prefixes, we will conduct prediction algorithm and decide which one can be loaded into the cache in the next stage. The other popular prefixes will remain in the cache. Normally the involved prefixes that need to be evaluated only account for approximately 10% of the cache size and the proportion will reduce as the cache size increases. In our experiment, the incremental update process may predict 1000–3000 candidate popular prefixes and only takes less than 10 s on a PC with a Pentium III 450 MHz CPU.

Theoretical analysis can be derived from the incremental update process. Suppose the $X_0$ series has $m$ traffic records. The complexity of GM(1,1) algorithm is linear to $m$: $O(Cm)$. On predicting the traffic loads of all $N$ prefixes, the time complexity will be $O(CNm)$. However, our optimization reduces the overhead considerably. The incremental update process filters out most prefixes with trivial traffic on the last time interval. At each update interval, the prediction will be conducted on a few thousand candidate popular prefixes which account for about 10% of the cache size. All in all, the computational overhead of GM(1,1) will be $O(CNm)$, where $n$ is the number of candidate popular prefixes in the RIB. In our experiment, $n$ is about 1500–2500, $m = 10$, which is not a heavy burden to a modern router.

The third concern is the delay of ORTC FIB suppressing algorithm. In the RIB, a popular prefix may have some overlapped sub-prefixes. The ORTC algorithm may substantially reduce the number of loading sub-prefixes. The complexity of ORTC algorithm depends on the structure of the RIB itself ([25] evaluates the overhead of ORTC). In ORTC, a popular prefix has been set as the root of a binary tree and its sub-prefixes are either intermediate nodes or leaves. The ORTC algorithm requires traversing the tree 3 times (in [20] an improved version only has 2 passes of the tree). In the worst case, suppose the height of the tree is $h$ (which is no more than 16), a traverse has to visit $2^h$ nodes. The overall computational complexity is $O(3 \times 2^p)$, here $p$ is the number of overlapped popular prefixes. However in most cases the ORTC tree is far from a full tree, mostly it has much fewer nodes than a full tree. In a real BGP route table, there are only a few popular prefixes that are heavily overlapped by sub-prefixes. And the number of intermediate nodes that need to be visited can be much less than $2^h$. In our simulation, the calculation of FIB suppressing for all overlapped popular prefixes only takes a few seconds when processed by a Pentium III 450 MHz CPU. Modern routers have CPUs that will outperform this rather slow processor. What’s more, ORTC is optional in our method. Applying ORTC is to make more room in the cache and consequently to improve cache hit rate. Other FIB aggregation techniques can be applied as well.

The last concern is the overhead of cache update. The loading of popular prefixes and/or BGP route update should not interrupt forwarding, because we propose a design of standby cache. The popular prefix loading and route update only involve the standby cache, and the cache replacement can be smoothly conducted by switching from one cache memory to another. Although this mechanism requires more physical memory space, it is worthwhile to do so, since the cached routes only account for approximately 5% of the RIB. The general memory consumption can be an order of magnitude lower than that of the status quo forwarding mechanism. Suppose a modern router can maximally accommodate a RIB of 1 M prefixes. With the same hardware capacity, our method can load about 500 k cached popular prefixes that account for 5% of a RIB and consequently the maximum RIB size that can be supported in our route cache mechanism is approximately 10 M entries with affordable forwarding delay. A routing table with 10 M entries is approximately 27 times larger than a full BGP table today. In other words, 10 M represents 27 times more networks than today’s entire public Internet.

All in all, in the scenario of the modern BGP routing, the traffic trace collection requires about 10 MB in a relatively slow memory; the prediction process will take no more than ten seconds with the incremental update process and FIB suppression algorithm will take another a few seconds optionally. The standby cache memory will be ready on time for cache replacement. The route update only involves the updated popular routes. An ORTC recalculcation of the updated popular prefix will not introduce significant delay because most of the cached routes will not change.

Being verified with different working Internet traffic traces, our method can dynamically cache 5% of BGP RIB in the FIB, and the cache miss rate is less than 5%; the ideal
7. Conclusion and future work

Our study gives an in-depth research on the features of the Internet traffic distribution across BGP prefixes. The traffic load over BGP prefixes has a power law distribution. Some of the prefixes are more popular than others, which makes the route caching mechanism justifiable. This paper studies how to catch the popular prefixes and minimize the cache miss rate at reasonable costs. Our concern focuses on how to catch the unstable popular prefixes efficiently and ameliorate the side effect of loading their non-popular sub-prefixes effectively.

Our method applies a grey model prediction and the FIB aggression technique. The GM(1,1) model can retrieve information efficiently from limited traffic traces (sampled packets) and adaptive to the bursts over non-popular prefixes very well. The ORTC algorithm can suppress the number of sub-prefixes effectively when loading a popular prefix into the cache.

The evaluation based on simulation with real traffic traces shows that our method outperforms other caching strategies, and its performance is close to the ideal limits. The theoretical analyses and experiments prove the general overhead of our method is acceptable.

In general, we have the following findings:

- The popularity of the BGP prefixes varies; and the long term stable popular prefixes only account for about 1–2% of the RIB and contribute approximately 80% of the traffic loads.
- There exist many short-term stable popular prefixes which can only be caught by dynamic cache strategies.
- Cache-miss-based strategies are not effective due to their high replacement overheads and their performance is influenced by the bursts of some non-popular prefixes.
- Prediction-based cache replacement method is adaptive to the dynamics of popular prefixes and outperforms static and cache-miss-based strategies at all cases of our experiments.
- Our prediction-based strategy can catch most of the popular prefixes when set the referring history with a proper range and retrieve information efficiently from the corresponding traffic traces.
- Overlapped sub-prefixes of the popular prefixes may contribute trivial traffic. This problem can be one of the major challenges of route caching mechanism. Optionally applying effective FIB suppression techniques can ameliorate this problem and finally make route caching more efficient.

In the future, we will try to acquire real traffic traces from more representative ISPs, for example, the tier-1 ISPs which would provide massive transit traffic to numerous edge ASes. Their production traffic may enhance our understandings on the features of popular prefixes.

We are expecting more comprehensive theoretical analyses in the future to explain the relationship between the BGP routes and the traffic loads on them. This kind of research may help to understand the Internet traffic dynamics and hopefully will also be useful on designing new scalable Internet routing systems/schemes.

Acknowledgement

This work was supported by the Key Project of Chinese National Programs for Fundamental Research and Development (973 program) No. 2009CB320501 and Specialized Research Fund for the Doctoral Program of Higher Education (SRDFP) No. 20080030034.

References


Wei Zhang received his B.A. degree in University of international relations, Nanjing and his M.S. degree from Wuhan national defense information academy. He is pursuing his Ph.D. in Tsinghua University. His research concerns Internet routing scalability and future Internet routing architecture.

Dr. Jun Bi received the B.S., M.S., and Ph.D. degree in computer science from Tsinghua University, Beijing, China. He was a postdoctoral scholar of High Speed Network Department in Bell Laboratories Research and a research scientist of Bell Labs Research Communication Science Division and Bell Labs Advanced Communication Technologies Center. Currently he is a Full Professor and Director of Network Architecture & IPv6 Research Division, Network Research Center, Ph.D. Supervisor of Department of Computer Science, Tsinghua University. His research interests: Network Architecture, Internet Routing and Addressing, IPv6 and Future Internet.

Jianping Wu received his B.S., M.S., and Ph.D. from Tsinghua University. He is a Full professor and director of Network Research Center, Ph.D. Supervisor of Department of Computer Science, Tsinghua University. From 1994, he has been in charge of China Education and Research Network (CERNET). He is a member of Information Advisory Committee, Office of National Information Infrastructure, Secretariat of State Council of China, and is also a vice president of Internet Society of China (ISC). His research interests include next generation Internet, IPv6 deployment and technologies, Internet protocol design and engineering.

Baobao Zhang received his B.S. from Beijing Jiaotong University. He is a postgraduate student in Tsinghua University. His research interests include network measurement, network performance and other Internet-related research work.