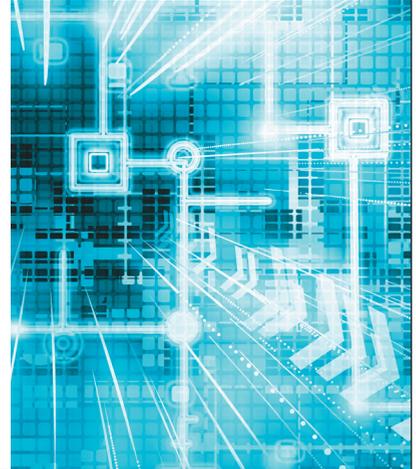


# Are We There Yet? User-Centered Temporal Awareness

➔ **Matt Adcock, Jae-Woo Chung,  
and Chris Schmandt**  
MIT Media Laboratory



Wearable devices could collect time-related information from the items and environments we interact with and create a personal temporal model.

**T**ime can be a precious commodity, and we often put great effort into deciding how best to spend it. Think, for example, about the decisions you might have made before choosing to read this article: Will I finish it before I need to attend to something else? If I don't have enough time to read it thoroughly, can I still get the gist by skimming it? If I have more time available, should I read a longer article instead?

It's not unreasonable to think that the technologies we use in our daily lives could help us make these sorts of decisions. For example, analysis of computer usage patterns can reveal our probable availability to colleagues. But, even better, it should be possible for devices to make some of these decisions for us while still leaving us with a comfortable level of control. We refer to such systems as exhibiting *temporal awareness*.

One way to realize temporally aware computing in everyday use is via wearable devices. Such devices could collect time-related information from the items and environments we interact with and create a personal temporal model. They could then use this information to change our behavior.

An example of temporal awareness in action, the AreWeThereYet? Player is a digital audio player that can compose a program of audio media likely to fit within the user's available listening time. AWTY estimates this time using the listener's current location and predicted destination as well as some knowledge of previous journeys.

## TEMPORALLY AWARE MEDIA PLAYER

Consider the following scenario. Kayla works at a travel agency and usually walks to work each morning. She's a little behind in listening to her podcasts, so she flicks on her AWTY. The player connects wirelessly to her phone, which is tracking her location and providing estimates of her remaining trip time. The AWTY subsequently assembles some of her latest podcasts into a program similar in duration to her journey.

While Kayla walks, the AWTY monitors her progress and automatically adjusts the playlist and play speed based on her phone's successive estimations of her arrival time. She can skip tracks, scan forward and back, and pause playback—the device simply recalculates a new playlist in the background. As Kayla arrives at

her shop, the podcasts finish playing just a couple of meters from the front door, and her AWTY goes to sleep. It's like clockwork.

As closing time rolls around, it starts to rain. Kayla calls her brother Jake to see if he can give her a ride. He can, but she'll have to wait for about 20 minutes. Kayla figures that, while waiting, she has enough time to get through a couple of chapters of the latest audio book.

Kayla tells her phone to connect to her brother's and provide his estimated arrival time to the media player. She then kicks back, knowing that she needn't worry about her brother's arrival interrupting a chapter of her book.

## TEMPORAL CONTEXT FROM LOCATION

To identify the available listening time, AWTY receives regular wireless updates from the user's GPS-enabled phone containing the estimated travel time (ETT) for the current journey. The phone calculates this ETT with a prediction application that uses a personalized model of possible routes.

For the application to detect a given route, the user must first train the system by traveling along that route, ideally as part of a normal daily

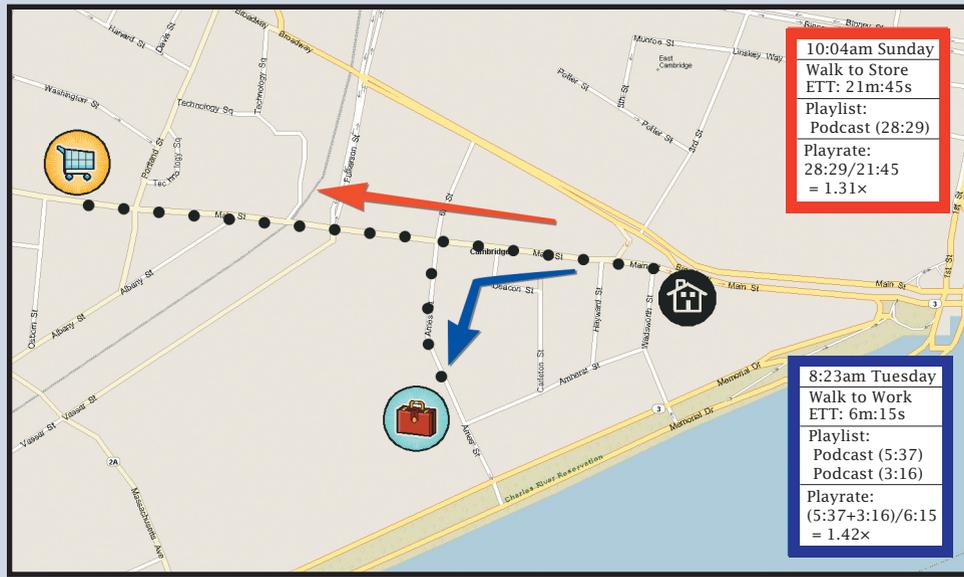


Figure 1. Example of two learned route templates for a particular user. One corresponds to a walk from home to work (blue arrow) and the other to a walk from home to the supermarket (red arrow). Black dots represent iPoints. When the user sets off from home, AWTY uses the time of day and day of the week to identify which ultimate destination is most likely, then estimates the remaining journey time. The text boxes show example playlists and their respective time-compression rates.

routine. Then, when the user revisits a given route, the application can predict the user’s ultimate destination as well as the associated ETT.

AWTY collects information about intermediate points every 50 meters with GPS fixes and uses these iPoints to generate a route template, as Figure 1 shows. The template for any given route also contains information about the time of day when the user was on the route.

The prediction of the ultimate destination is based on the user’s current direction, route traveled so far, and time of day. The ETT calculation is based on the user’s current speed as well as average speed during previous traversals of the same route.

Each route template is divided into regions of similar speeds roughly corresponding to different modes of transport—this is useful for the person who walks some distance then, say, catches a bus. For each region, the route template maintains a weighted average of the speeds collected during previous journeys along the respective route, giving the most weight to

the most recent journey. AWTY uses a given template to calculate the ETT whenever a user’s current travel speed is close to the corresponding weighted-average speed in the template and the time of day is similar to the times at which that template was previously recorded.

On average, the algorithm updates its prediction every 30-40 seconds. When the route prediction application updates, it sends the new ETT via Bluetooth. The ETT message is a simple string containing the estimated number of seconds to arrival.

### TIME-MATCHING TECHNIQUES

There are two main ways that a media system such as AWTY can play audio so that it fits within a specific time window: *time compression* and *track selection*. These techniques aren’t mutually exclusive.

#### Time compression

Essentially, time compression “squashes” a selection of audio tracks into the anticipated time window. This

works pretty well for most spoken word audio but faces hurdles where music is concerned. Several well-known techniques permit speech to be played at increased speeds while maintaining the original pitch.

Speech playing at around 1.4x normal speed is usually highly comprehensible. Faster speeds up to 2x normal speed are sometimes acceptable but often require additional concentration. People quickly adjust to time-compressed speech, and in fact, after a short period of listening, some prefer it to uncompressed speech. AWTY lets users set—and forget—a comfortable maximum time-compression rate.

The AWTY prototype’s initial implementation focused on audio books. In this case, the track (chapter) order is predetermined and the need for time compression greatest. The only possibility of track selection is to determine how many of the audio book chapters will be played. We accordingly developed an algorithm that both selects how many tracks to play and at what rate to play them.

The system first determines the maximum number of tracks—including any partial current track—that can be played within the remaining estimated listening time and adds those tracks to the playlist. This involves simply adding track lengths and dividing by the user's preset maximum time-compression threshold.

The next step is to divide the estimated amount of listening time by the total normal-speed play times of the selected tracks. This gives the play rate needed for the tracks to play until the estimated end of the journey. Figure 1 shows example playlists and their respective time-compression rates.

The system uses the resulting play rate or time-compression factor until either the estimated remaining listening time diverges from the remaining play time or there is user interaction, such as rewinding. If either of these occurs, it simply recalculates the play rate.

### Track selection

The second time-matching technique is to simply choose tracks from an audio library with play lengths that add up to the required time. This relies on a large source of audio material and can be used with music or unconnected recordings of speech. The experience for the user is similar to activating “shuffle” or “random” on today's commercially available media players.

One way to calculate a playlist that is as close to the estimated listening time as possible is to exhaustively solve the knapsack problem (named after a hypothetical thief trying to choose a set of objects that best fills his knapsack).

However, assuming the user has a suitably large set of tracks from which to draw and that the given journey is an order of magnitude larger than the average length of tracks in the library, we can employ a cheaper, approximate algorithm:

[This only needs to be done once.]

0. Create an index of the entire collection ordered by track length

and determine the average track length.

1. Randomly select individual tracks and add them to a list until the list is close to the total available listening time.
2. Rank the selected tracks by their difference from the collection's average track length and assemble a playlist such that the longest and shortest tracks will play first.
3. Swap some small number of tracks near the end of the list—the ones closest to the average length—with tracks that will result in the total play time equaling the available listening time. This last step is possible in typical popular music collections that exhibit a normal distribution.

As the user's journey progresses, the system further refines the estimated available listening time. It can then swap the latter song(s) in the list accordingly. This is possible because the system doesn't disclose the list to the user.

### THE FUTURE OF “TIME/TRAVEL”

Temporal awareness is an area of human-computer interaction that has remained relatively underexplored. This is somewhat surprising given the impact of time on our lives and the untapped temporal information that already pervades everyday technology.

In the AWTY Player, we use learned personal journey histories to generate temporal context information. However, many other potential sources of real-time temporal information already exist.

Most car navigation systems calculate an estimated journey time, as do navigation systems on many trains and airplanes. The Massachusetts Registry of Motor Vehicles provides expected waiting times for each of its branches on the Web. Some parking garages include sensors in every bay

and could estimate time-to-park from the rate at which cars enter the garage and at which bays are filled.

An opportunity exists to develop a computer-readable interchange language for representing temporal information—perhaps in XML, similar to RSS. Then, everyday technologies such as photocopiers and subway trains, as well as more complex temporal prediction models, could use this language to report how many seconds are left until the current task is completed, the destination is reached, or some other key event occurs.

Ultimately, temporal context information could also be shared anonymously online and then harvested from people who have been in similar situations.

**I**t is our hope that temporal awareness will receive more attention from researchers and technologists. In the not-too-distant future, we expect our kids to be playing a computer game with the full confidence that they will run out of lives at the exact moment dinner hits the table. 

*Matt Adcock was a research assistant and master's student in the Speech+Mobility Group at the MIT Media Lab and is currently a research engineer at Australia's Commonwealth Scientific and Industrial Research Organisation (CSIRO). Contact him at [matta@media.mit.edu](mailto:matta@media.mit.edu).*

*Jae-Woo Chung is a research assistant and PhD student in the Speech+Mobility Group at the MIT Media Lab. Contact him at [jaewoo@media.mit.edu](mailto:jaewoo@media.mit.edu).*

*Chris Schmandt is a principal research scientist and director of the Speech+Mobility Group at the MIT Media Lab. Contact him at [geek@media.mit.edu](mailto:geek@media.mit.edu).*

**Editor: Bill N. Schilit, Google;**  
[bill.schilit@computer.org](mailto:bill.schilit@computer.org)